The Complete Guide to

Application Performance Management Tools

by Henn Idan

Table of Contents

What's an APM and Why Do You Even Need It?

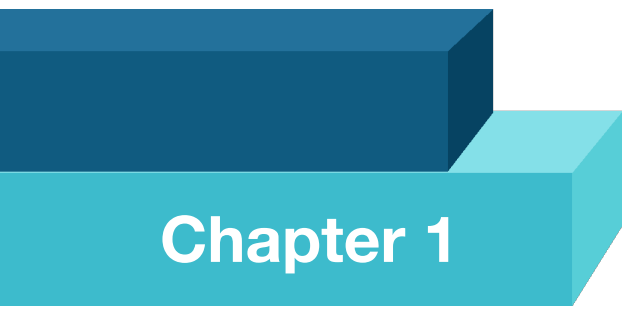
One of the common ways to get an inside view and monitor how an application performs and behaves in production, is through an APM tool. Application Performance Management has been around for a while, though it seems like many developers are not comfortable with it yet.

APM provides us with analytics around our application's performance – at the core this means timing how long it takes to execute different areas in the code and complete transactions – this is done either by instrumenting the code, monitoring logs, or including network / hardware metrics.

There are a lot of players in the APM space, each with its own features, offering and pricing. Some focus on SMB's, while others target enterprises and large companies.

That why we've decided to combine everything there is to know about these tools, big and small combined, and see which one is the best option for your monitoring needs.

Let the monitoring begin.



AppDynamics vs Dynatrace

Enterprise Application Performance Monitoring: A practical comparison between AppDynamics and Dynatrace

If you were to pick two tools that would appear on most enterprise APM (Application Performance Monitoring) evaluation shortlists it would be AppDynamics and Dynatrace.

Both didn't become leaders in APM overnight but did so through different journeys, despite having similar foundations in Java diagnostics. Dynatrace started more as a pre-production performance tool in 2005 for developers and QA testers, while AppDynamics was founded in 2008 with production use cases focusing more towards operations and application support teams. Today, pretty much both of them have expanded into each other's territories covering nearly all performance monitoring use cases an enterprise could want.

In the following post we will uncover the truth behind "Newspeak" (Or marketing speak) and help you understand what each tool offers, and which one best suits you.

If you want to monitor how your application performs and behaves in production, you need to use an APM to monitor the app performance. Gartner defines APM as five core components on which we've based our comparison.

Before drilling-down into both AppDynamics and Dynatrace it's important to understand the history behind each of the product sets as both vendors have evolved through several acquisitions and years of organic growth.

AppDynamics:

Company	Acquirer	Price Paid	Year	Product Name Today
AppDynamics	-	-	2012	AppDynamics End User Monitoring
DBTuna	AppDynamics	undisclosed	2013	AppDynamics for Databases
Nodetime	AppDynamics	undisclosed	2013	AppDynamics APM
BugBuster	AppDynamics	undisclosed	2015	AppDynamics APM
AppDynamics	-	-	2015	AppDynamics Application Analytics
AppDynamics	-	-	2015	AppDynamics Browser Synthetic Monitoring

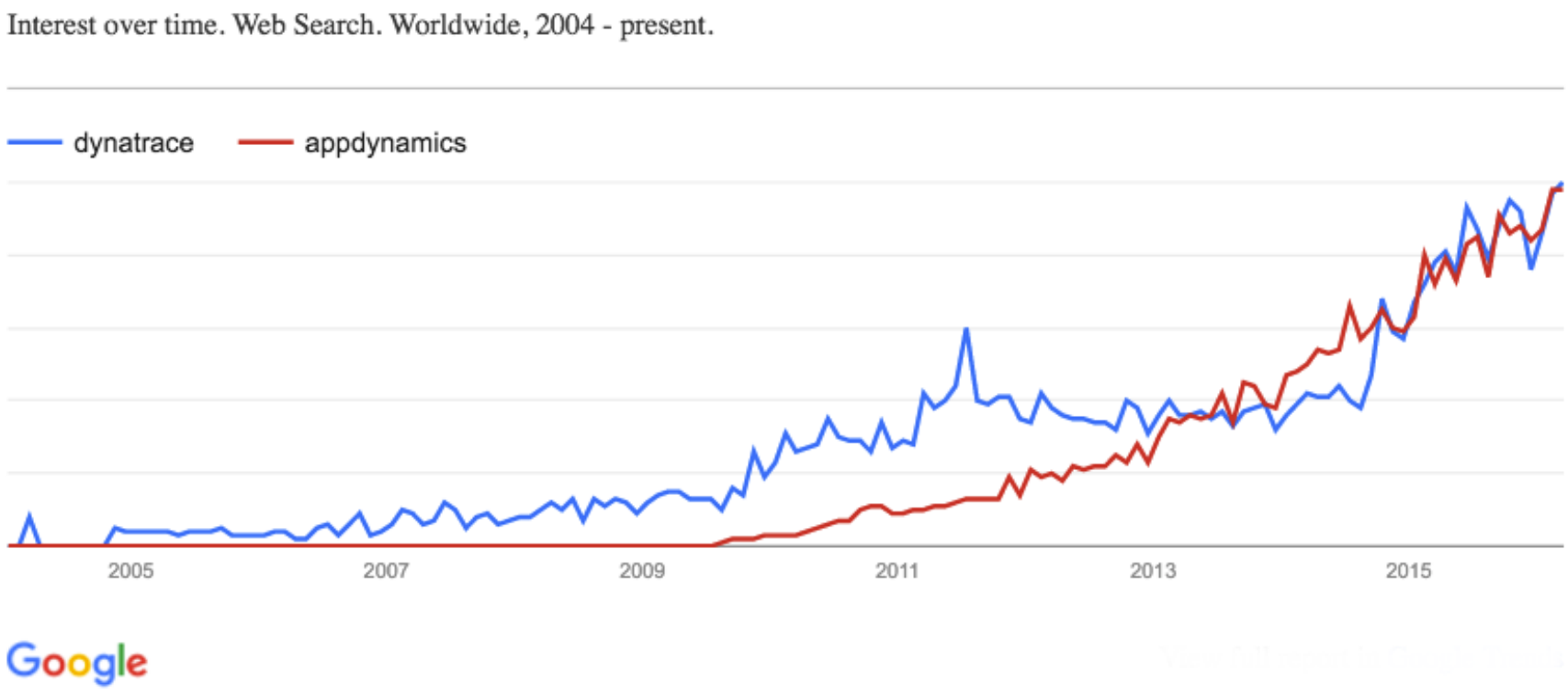
OverOps

Dynatrace:

Company	Acquirer	Price Paid	Year	Product Name Today
Adlex	Compuware	$36m	2005	Dynatrace Data Center RUM
Gomez	Compuware	$295m	2009	Dynatrace Synthetic Monitoring
Dynatrace	Compuware	$256m	2011	Dynatrace Application Monitoring (APM)
Dynatrace	-	-	2012	Dynatrace User Experience Mgmt
Keynote	Thoma Bravo	$395m	2013	Dynatrace Synthetic Monitoring
Compuware (Dynatrace)	Thoma Bravo	$2.5b	2014	Dynatrace now operates as a standalone entity outside of Compuware

OverOps

AppDynamics and Dynatrace are head-to-head competing over the users:



View full report in Google Trends

For the benefit of this comparison we've decided to take on the core on-premise APM offerings from AppDynamics and Dynatrace using Gartner's five dimensions. Dynatrace's APM is also featured in a separate SaaS product called Ruxit which was not covered in this post.

If you're interested in more APM tools, check out these 15 useful tools for production environments.

1. Features

1.1. End User Experience Monitoring

The first dimension of APM is End User Experience Monitoring (EUEM). It's a term used to describe the multiple approaches to monitor what an end user might be experiencing.

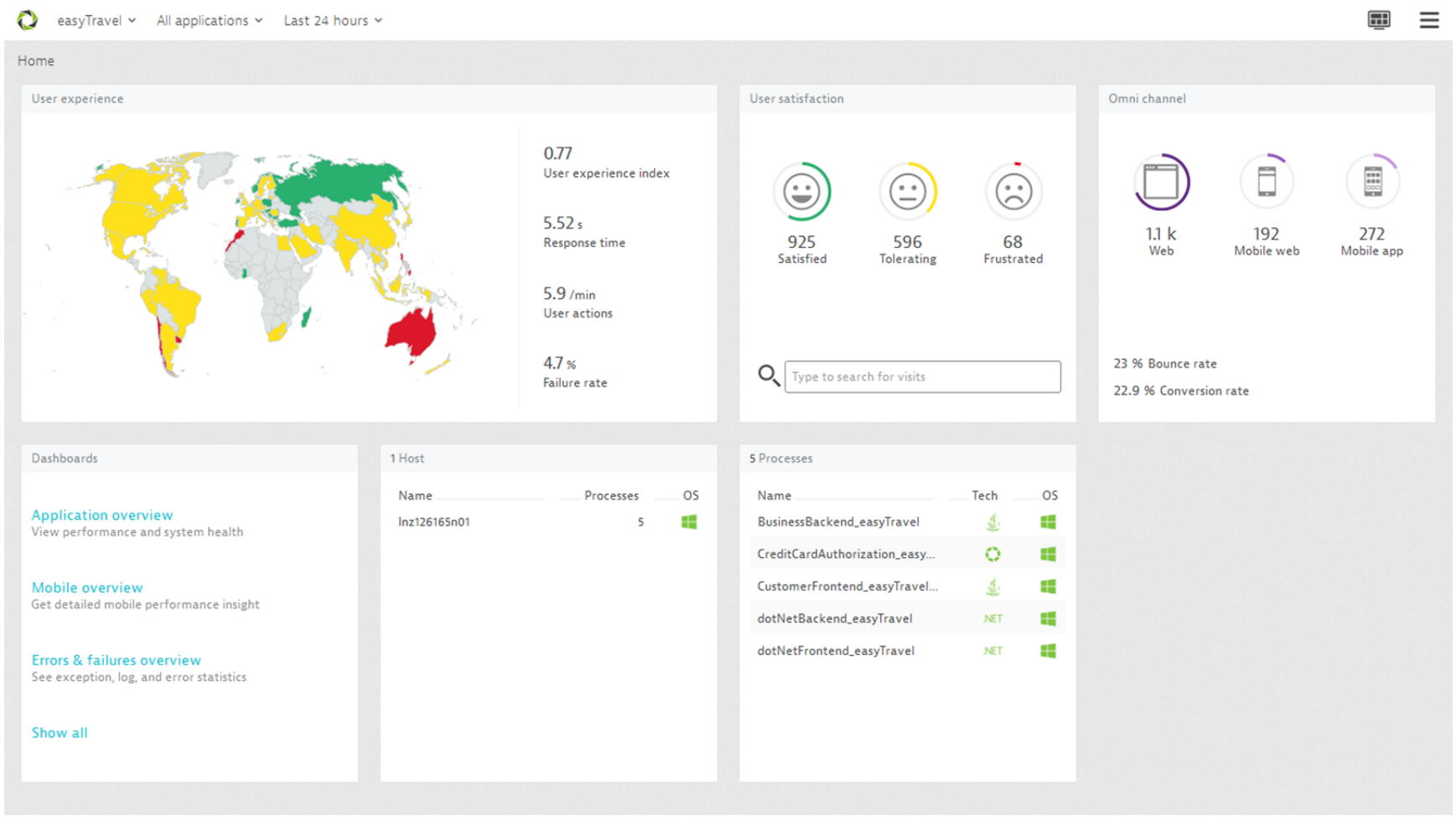
Dynatrace: The company divides its EUEM capabilities into 3 products: "User Experience Management", "Synthetic Monitoring" and "Data Center RUM (Real User Monitoring)". Don't be scared of the names, they are meant to sound explosive and expansive while they simply monitor the user experience from different perspectives.

The first tool, “User Experience Management” offers a browser-based JavaScript injection and SDK approach for mobile apps that captures metrics related to user requests and sessions, across different browsers and devices. For example, you can search for specific user actions in order to get a history of their performance / how long did they take to execute.

The “Synthetic Monitoring” service displays stats from different locations around the world. It’s also helpful for simulating test executions, comparing performance across problematic behavior patterns, monitoring host availability and third party services or even simulating high volume traffic.

“Data Center RUM” is the last solution which offers an agentless approach to end user monitoring by sniffing incoming HTTP requests (and other protocols) so it can piece together the latency of end user requests without needing to explicitly instrument web pages or native mobile apps.

To sum up: JS injection / SDKs, simulating visitors from different locations and network activity.



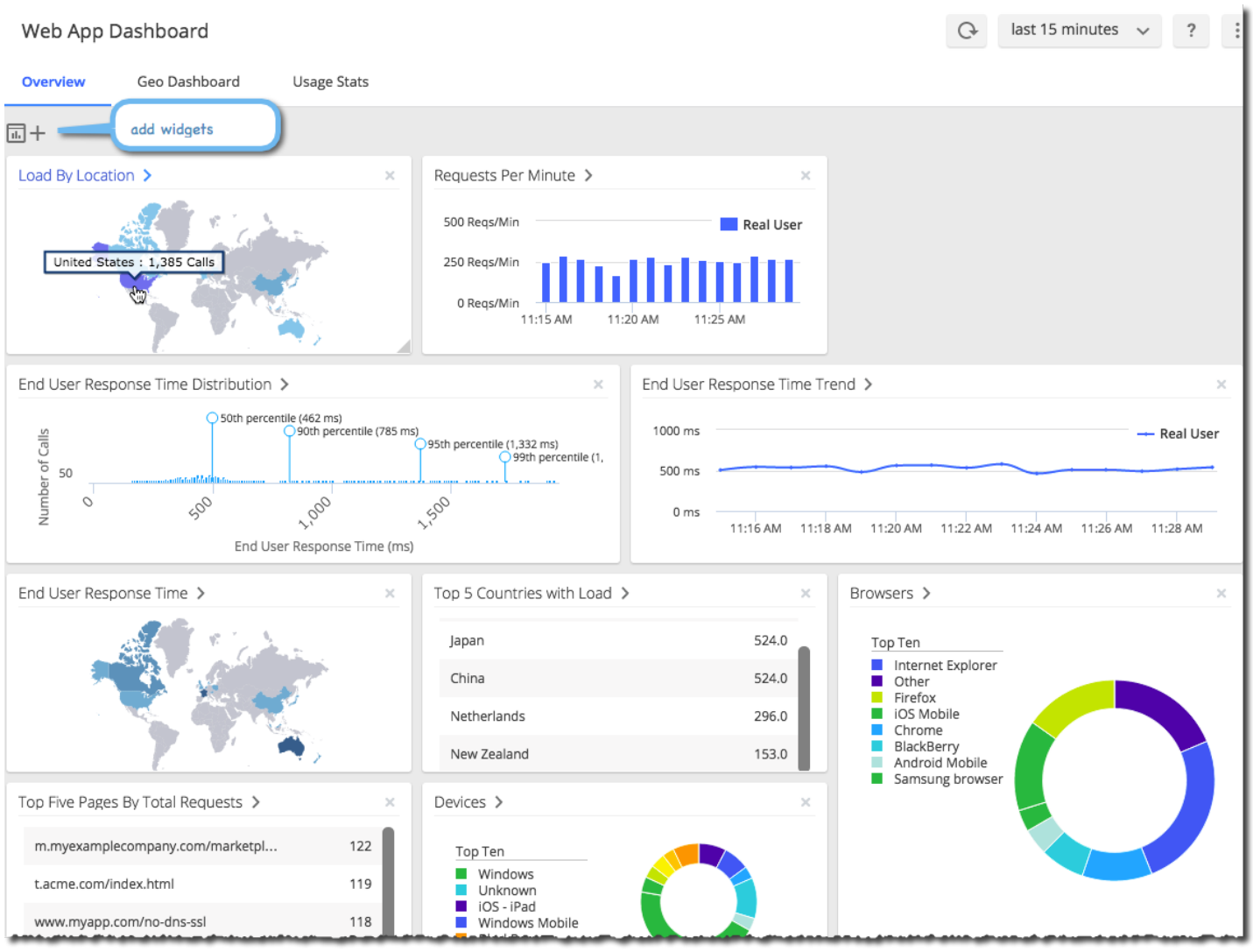
The Dynatrace User Experience Management Dashboard

AppDynamics: The “End-User Monitoring” category also expands into 3 different products: “Browser Real-User”, “Browser Synthetic” and “Mobile Real-Time” monitoring. In simple terms, you’ll be able to monitor user experience and interactions and benchmark your performance from specific regions.

“Browser Real-User” is a JavaScript injection approach that allows you to capture the experience of end users from the browser. You’ll be able to reproduce your users’ journey inside the app, see individual transactions, and identify performance issues.

“Browser Synthetic” will monitor the availability of your app 24/7 and will report its latency around the world. It simulates traffic from browsers to assess the actual end-user experience. You can also retest conditions in order to eliminate erroneous results, or confirm them.

“Mobile Real-Time” is all about native mobile apps, offering the same measurements and options, only across mobile apps and devices. It allows to triage the steps the user took before the app crashed, view latency across services as well as errors so you’ll be able to identify and troubleshoot issues. Instrumentation is done via standard SDKs for iOS and Android.



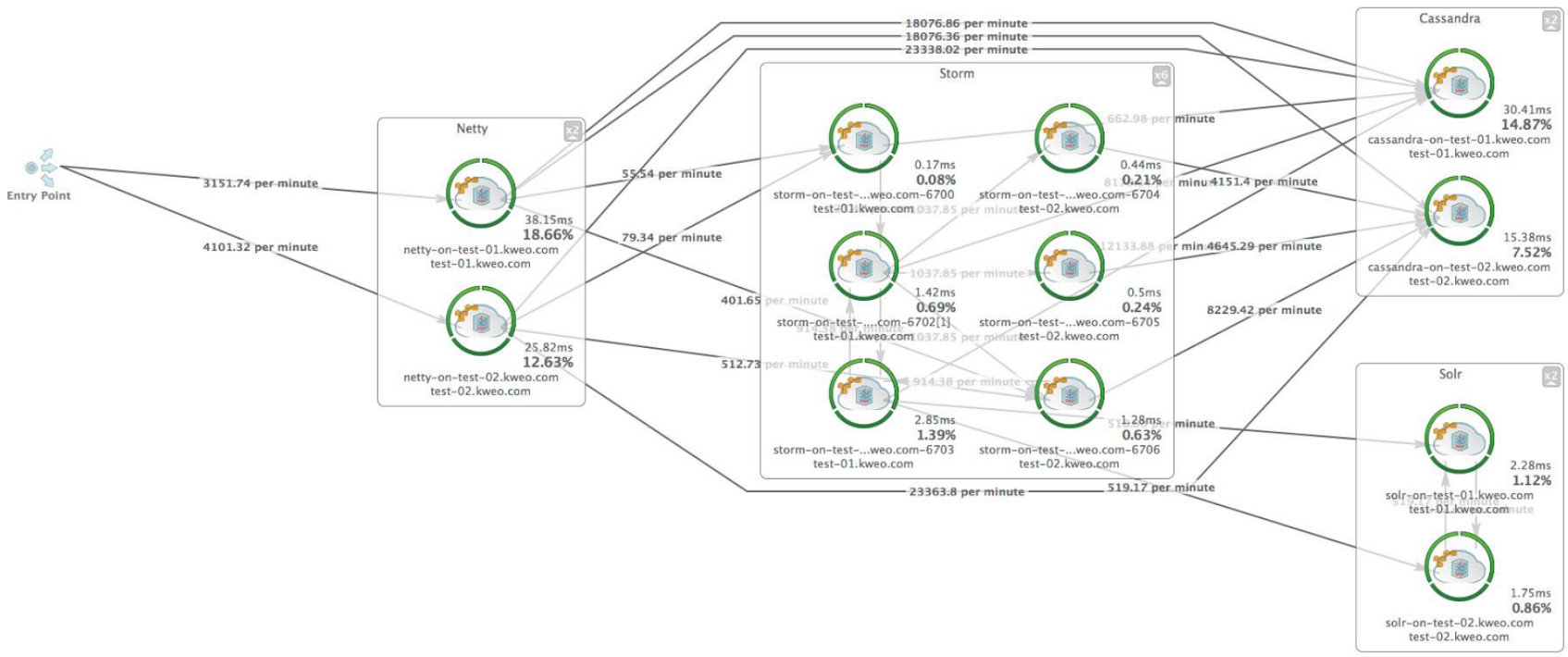
The AppDynamics Real User Monitoring Dashboard

Bottom line: Dynatrace has breadth and depth in this area through their various acquisitions of Gomez, Keynote, Adlex and organic growth but their products are less modern and unified than AppDynamics. For example, there is no single-pane-of-glass across User Experience Management, Synthetic Monitoring and DC RUM whereas with AppDynamics their products are less mature but easier to use and navigate.

1.2. Runtime Application Architecture Discovery Modeling

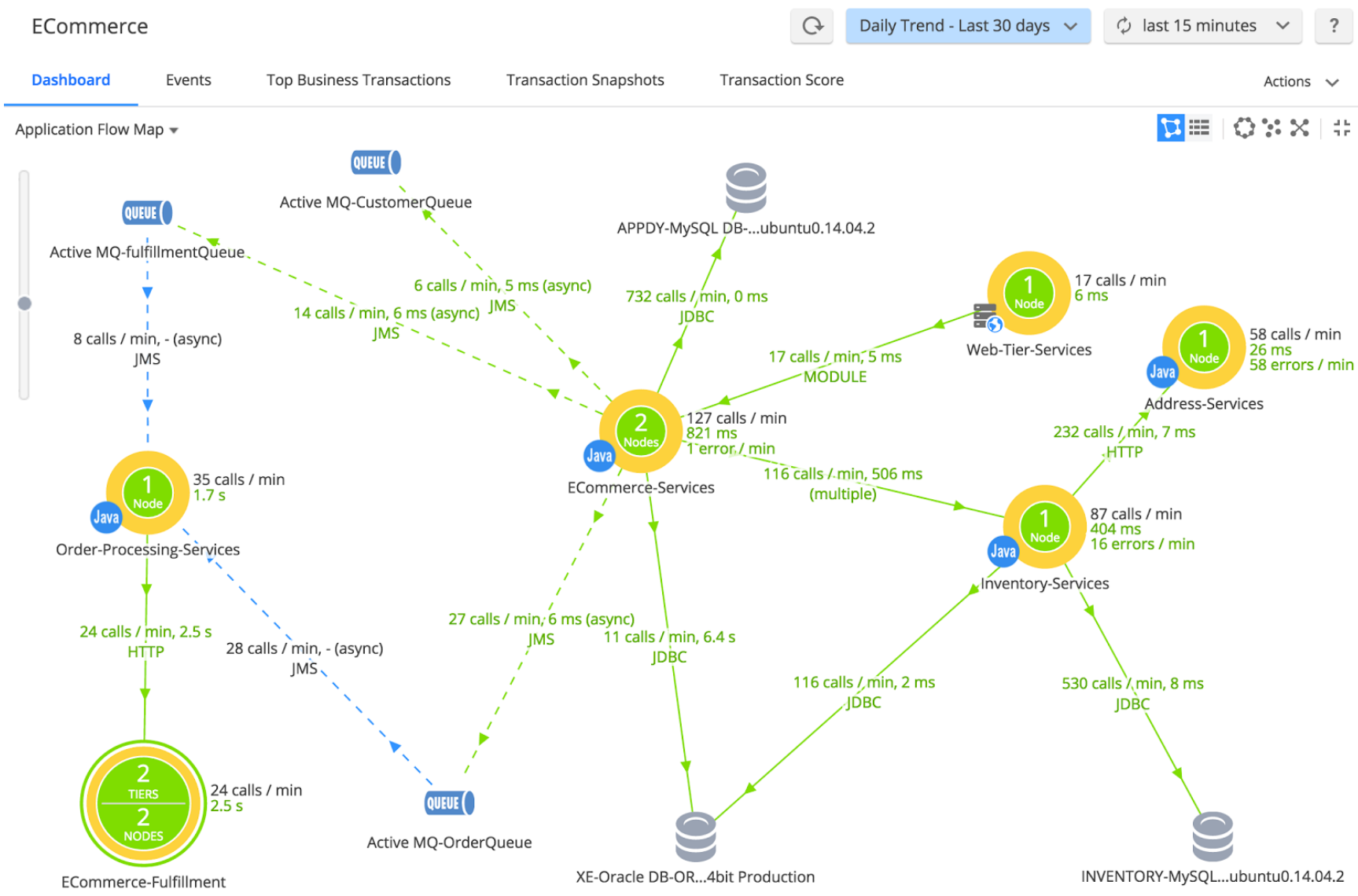
This is perhaps the coolest dimension of APM where you get a live topology map of how end user requests traverse the many application run-time environments. For example, in a microservices architecture, a single user transaction could touch up to several hundred different components and servers, with each hop potentially contributing latency. With topology maps, you should be able to understand where the latency is spent for the application.

Dynatrace:



The Dynatrace topology map

AppDynamics:

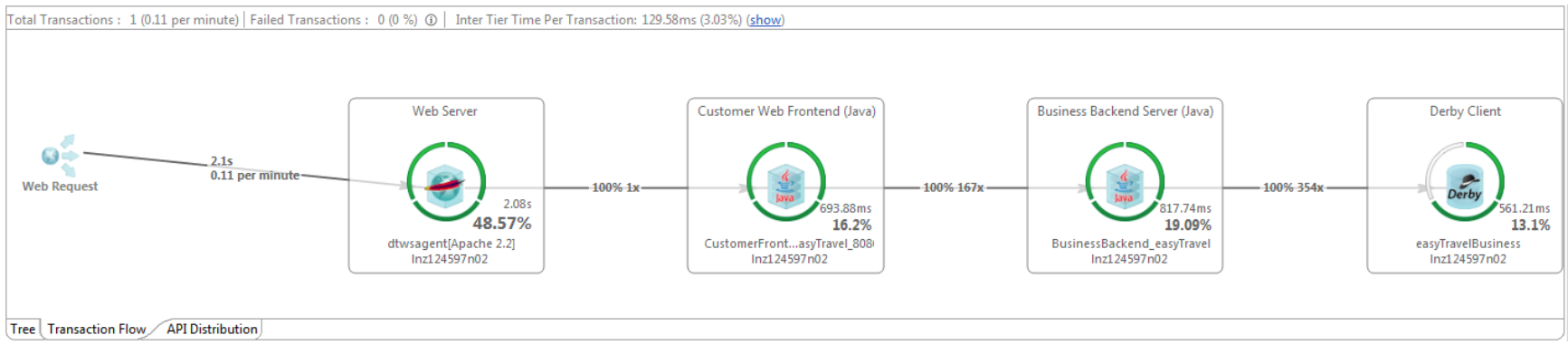


The AppDynamics topology map

Bottom line: As you can see, both Dynatrace and AppDynamics have similar application architecture modeling capabilities.

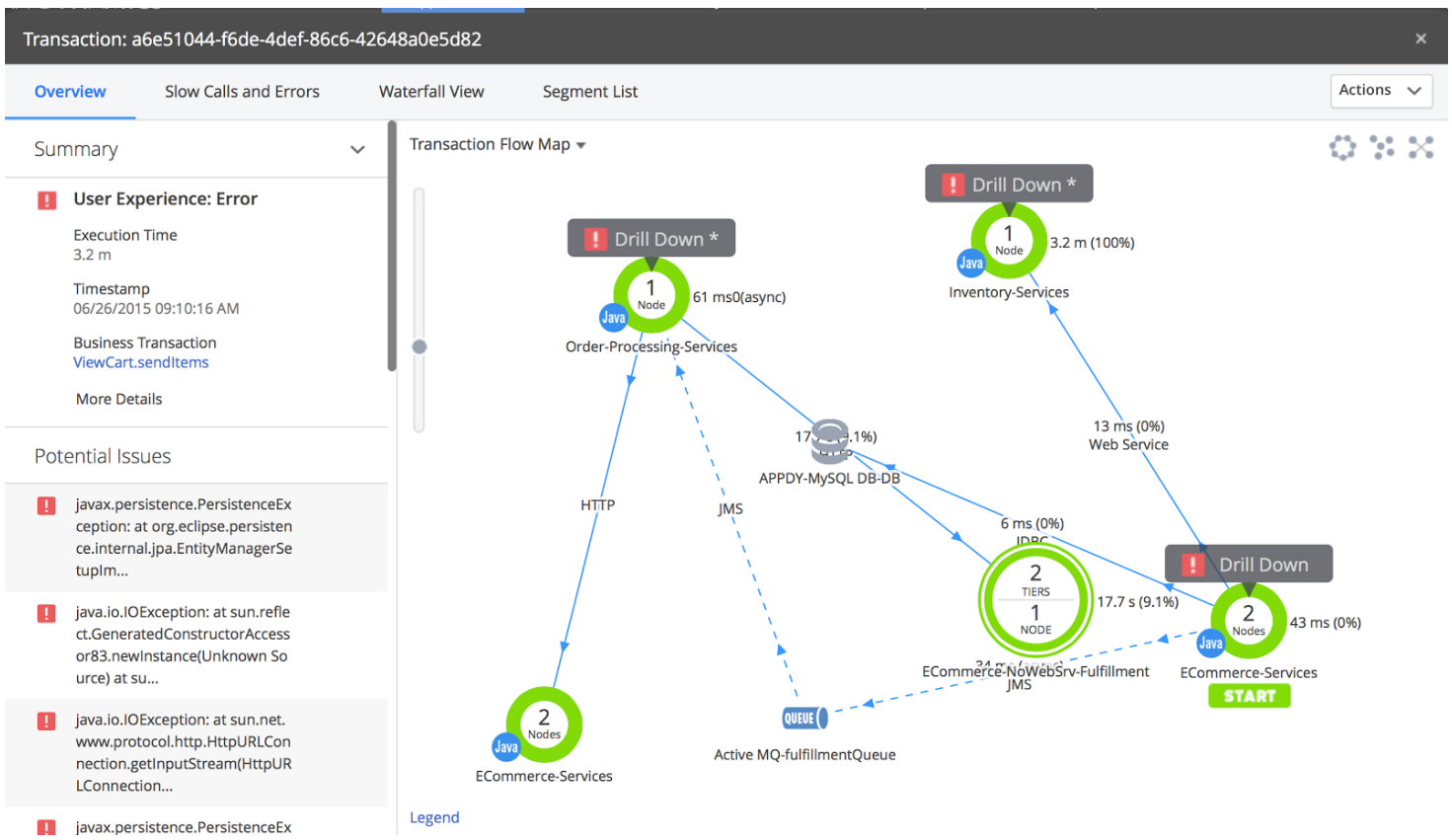
1.3. User-Defined Transaction Profiling

Dynatrace: Captures all user requests with limited stack traces (that include only the topmost elements). These profiles are known as “PurePaths” which visualize the journey of every request. Transaction profiling has always been a strength of Dynatrace, and they were one of the first vendors to trace transactions across heterogeneous run-time environments (e.g. hybrid JVM and CLR architectures). Dynatrace provides multiple drill-down options from a given PurePath so it’s possible to slice and dice data from different angles.



The Dynatrace transaction dashboard

AppDynamics: Captures business transaction anomalies with complete stack traces. Similar in a lot of respect to Dynatrace but perhaps more intuitive without the need to frequently drill-down/rotate views. The rest of the available information is automatically bubbled up to you, so it’s visually easier to understand and act on.



The AppDynamics transaction dashboard

Bottom line: While on the outside both vendors successfully profile business transactions, they do so in very different ways. Both solutions capture the response time of every transaction execution across distributed environments.

Dynatrace has designed its agents to capture limited stack traces for every transaction execution, whereas AppDynamics has opted for the anomaly diagnostic approach, meaning that more complete stack traces is only captured when a performance baseline or threshold has been breached. Each approach has its pros and cons depending on your monitoring use cases and requirements.

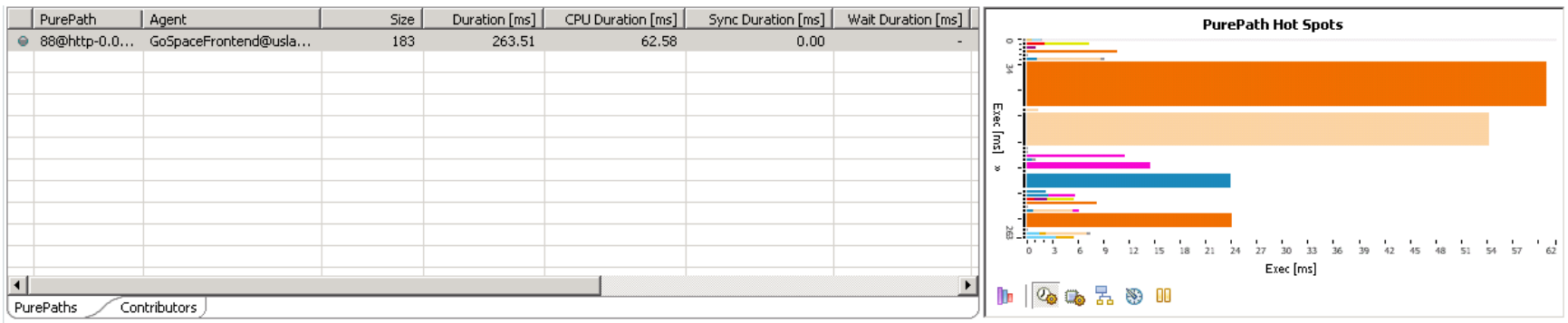
1.4. Component Deep-Dive Monitoring

Dynatrace: Captures stack traces and performance metrics for Java, Scala (if you're using Play framework with Scala, you're out of luck), .NET, PHP, Node.js, C/C++, JavaScript, iOS and Android applications.

Using a combination of byte code instrumentation and thread sampling it's possible to get stack trace data with method names individual user requests (as shown below). As previously mentioned, the stack trace data that Dynatrace provides is less complete than what AppDynamics provides.

This is due to the fact that Dynatrace collects stack trace data for every single transaction execution, therefore the use of instrumentation needs to be lighter so that overhead is controlled in production.

The look and feel of Dynatrace is very similar to that of an IDE so navigating stack traces and metrics is more user-driven, meaning you can pretty much customize whatever view you want using the dashlets inside Dynatrace.

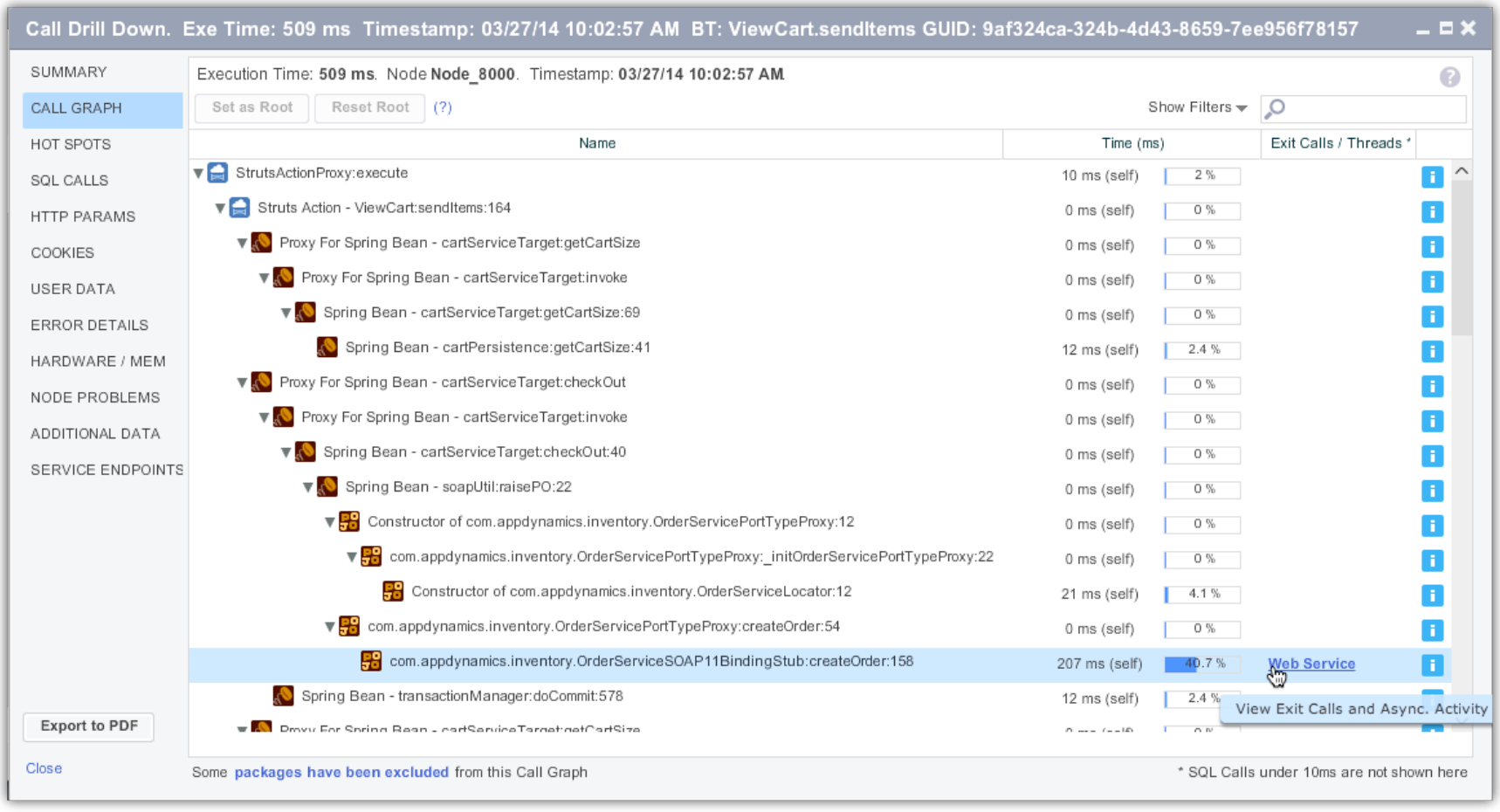


The Dynatrace analysis dashboard

AppDynamics: Captures stack traces and performance metrics for Java, Scala, .NET, PHP, Node.js, Python, C/C++, iOS and Android.

The first difference you'll notice is that the AppDynamics data is laid out in a more progressive/intuitive way, it also displays less data on screen at any given time. Instead of right clicking on data you can use the left-hand side navigation to switch contexts.

Stack traces are richer than what Dynatrace provides but the tradeoff is that you only get this data for anomalies.



The AppDynamics analysis dashboard

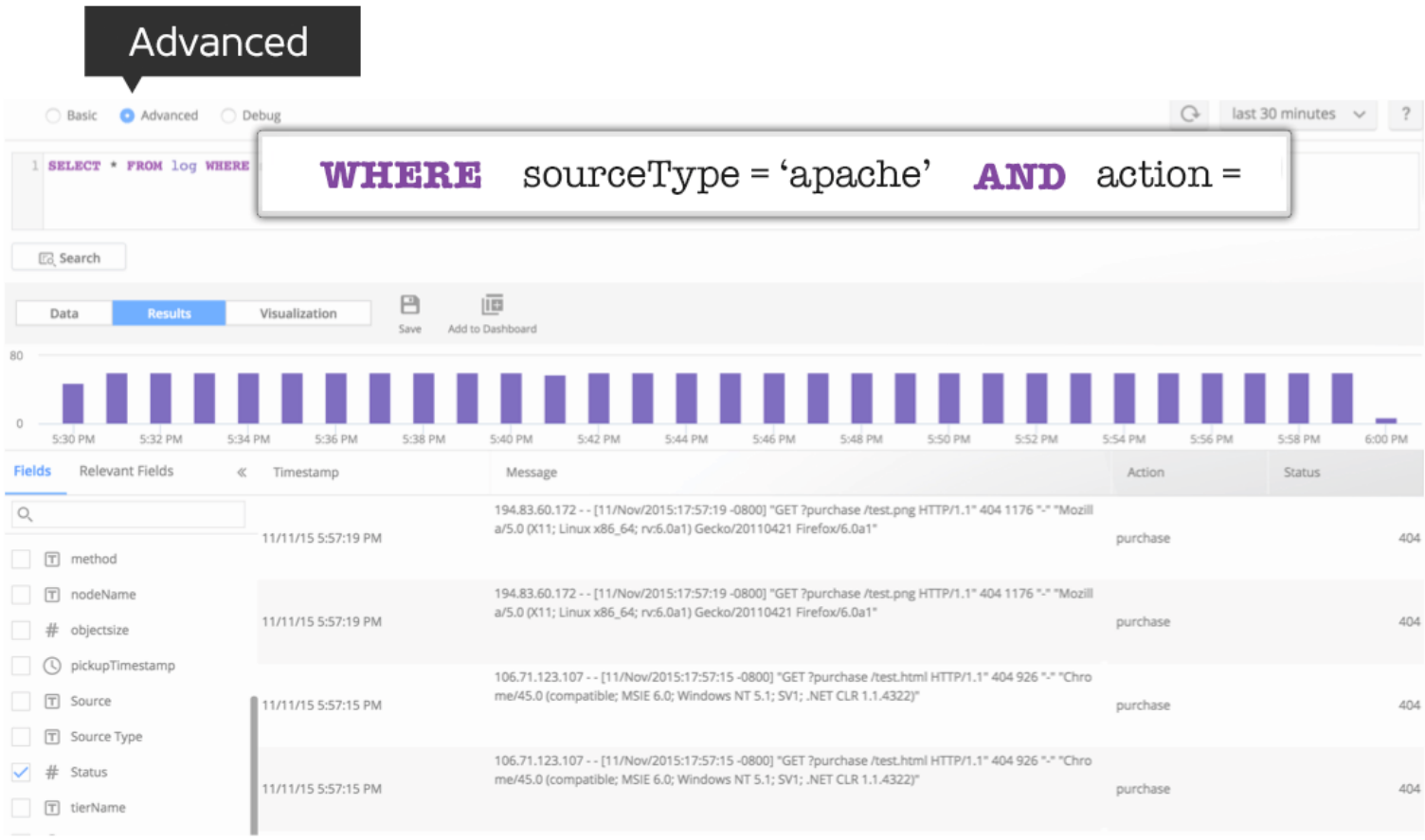
Bottom line: Honors are fairly even for both vendors in this category. Both tools were built to provide stack traces/metrics and this has become a core competency and differentiator against some of the other older vendors in APM like IBM, HP, CA and Microsoft.

You'll find similar levels of code diagnostics for the most common platforms (Java/.NET) with both tools but it's worth checking in advance if you're running on other programming languages like PHP, Node.JS and Python as stack trace data and metrics will vary.

1.5. Analytics

Dynatrace: New in Dynatrace 6.3 is the PureLytics feature. The PureLytics stream is optimized to send data in realtime into Elasticsearch for analysis, including user visits and actions, with Kibana's dashboard on top.

AppDynamics: AppDynamics has another product called "Log Analytics", a new capability that indexes and analyzes log files, similarly to Splunk and ELK. It collects structured and unstructured data, displays application issues and present errors that occur in real-time. You can search through these errors and create custom alerts or dashboards, to understand how errors impact application performance.



The AppDynamics Log Management Dashboard

Bottom line: AppDynamics provides log management features.

2. How to Solve the Errors You Find

While both Dynatrace and AppDynamics offer you performance monitoring features, they don't offer granular support when it comes down to runtime errors or exceptions when your application breaks. You'll be able to identify performance issues, understand how long it takes for a business transaction, page or request to load or even experience what your users are experiencing, but the lowest level of granularity you'll get to in production is the class and method names of where latency is spent.

Many teams who use an APM tool also choose to add OverOps to their monitoring stack. With both tools combined, you'll be able to identify errors, and view the source code related to the issue with the exact variable state at the moment it happened.

OverOps analyzes exceptions and log errors in production, and overlays the variable information from the moment of error over the actual source code of each method in the event's stack trace. This will give you the ability to analyze the exception or log error as if you were there when it happened.

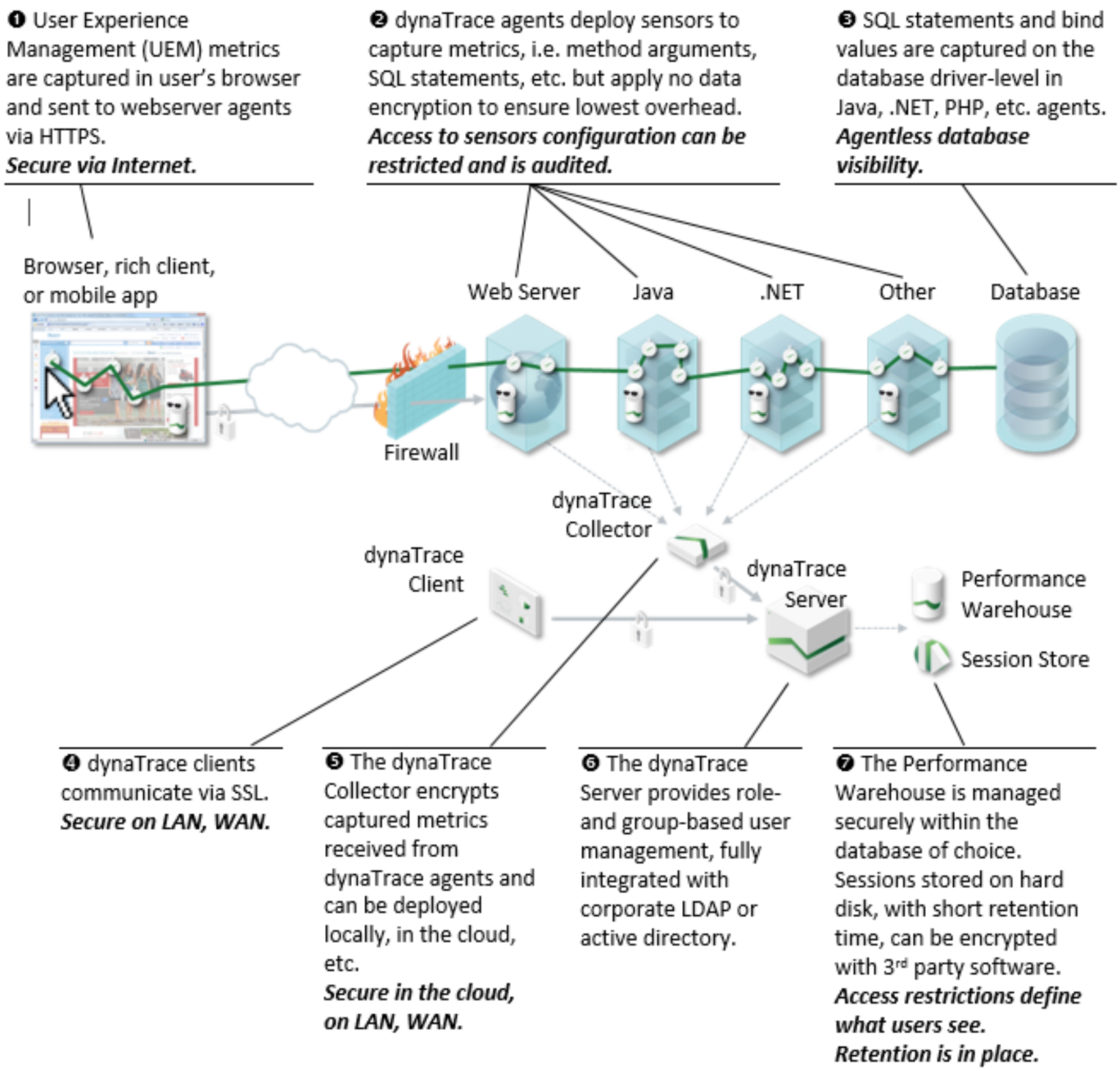
Bottom line: Whether you're using an APM tool or not, the only way to get down to the real root cause of each error is to get its full source, stack and state. If you're a Java or Scala developer, you should give OverOps a try.

3. Installation and Ease of Deployment

Dynatrace: We were actually quite impressed with the installation process, which is completely self explanatory. The company offers several installers for Windows, Linux and Mac OS, that contain everything – agent included. You can also run it all in a preconfigured Docker container.

After the installation you'll be asked to connect the agent with the server, and you'll be presented with video guides for each environment, if you'll need assistance.

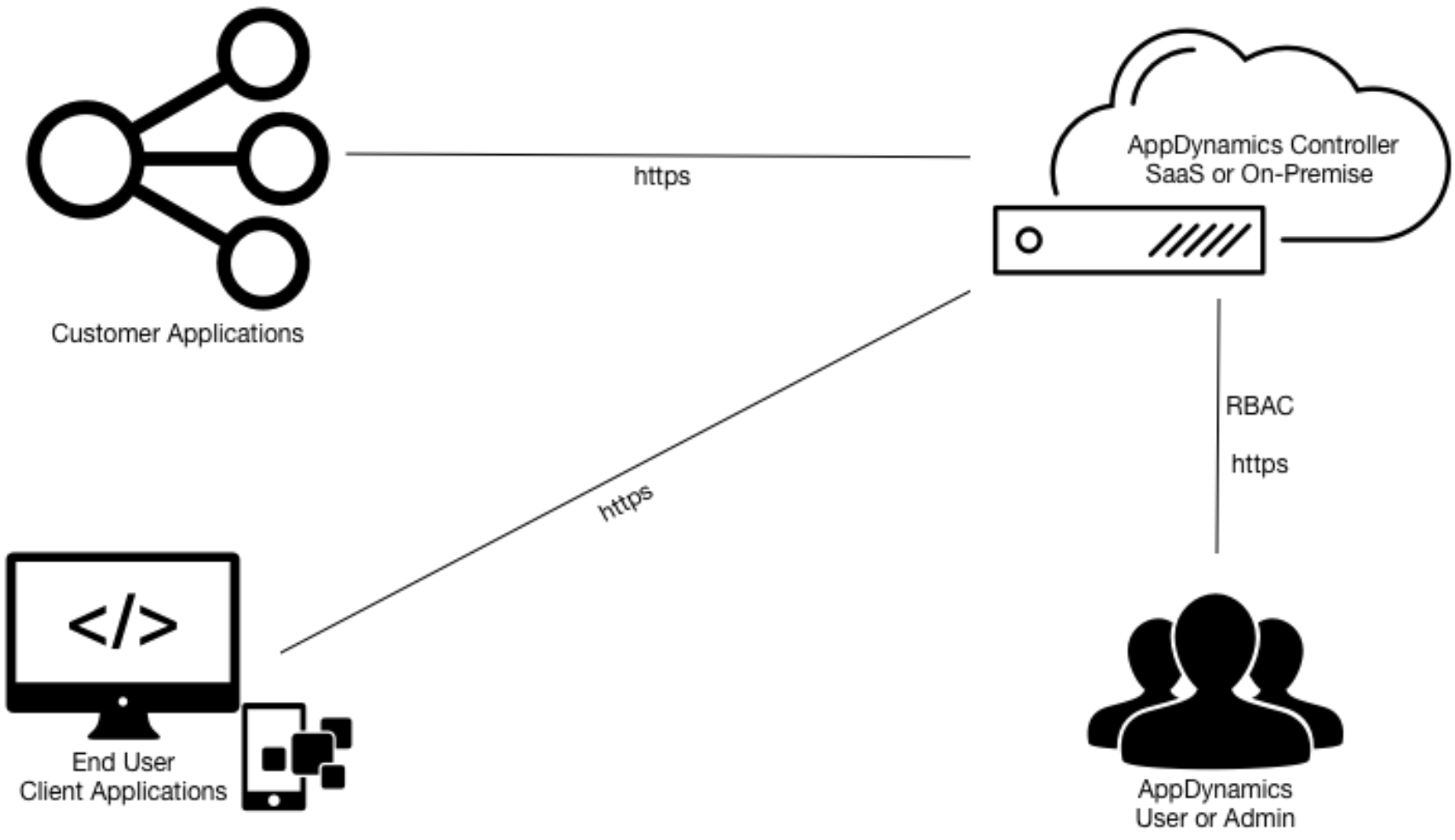
In large scale environments you will have to install multiple components so that Dynatrace can scale. For example, Dynatrace agents communicate with collectors which then communicate with the central Dynatrace server. You'll need to add more collector servers as the number of agents increases over-time.



Dynatrace architecture

AppDynamics: After signing up to the service, you are presented with an installation screen that has 4 steps and no further information. We looked for a download kit or instructions, but couldn't find them. It turns out that AppDynamics needs some time to set up your environment, and you can't start using it straight away. Although it makes sense, this was not explained on the main screen, and it took over an hour until we got our welcome email and full dashboard.

The AppDynamics architecture is simpler than Dynatrace to deploy as it only requires agents and a central management server known as the controller. AppDynamics collects and transmits much less data than Dynatrace so it doesn't require additional collector servers to scale. Although it requires an additional agent per machine for infrastructure metrics.



AppDynamics architecture

Bottom line: Both tools are relatively simple to get up and running. Due to the differences in data collection and transmission, AppDynamics is slightly easier and cheaper to scale for larger environments.

4. Dashboards and Ease of Use

Dynatrace: The company offers 2 dashboards: a local desktop client (that feels like a weird version of Windows 2000), and a web dashboard.

The web dashboard displays high level information, that includes user experience and satisfaction, your app channels (web or mobile) and the list of processes currently running.

If you'll find an issue or error and would like to analyze it or get more information about it, you'll have to switch to the desktop dashboard.

AppDynamics: The good news are that AppDynamic are no longer using a Flash based dashboard! The somewhat bad news are that the company still requires Flash support, since some parts of the dashboard still need it to tun. You have a full, detailed web based dashboard that offers an overview, and an option to drill down into the information you want.

Bottom line: Dynatrace 2-dashboards-setup could be a hassle for some, and it feels a bit outdated, especially if we need to switch between the two if we want to get more in-depth information about different scenarios. AppDynamics has a unified UI across all products/features but still has elements of Flash present.

5. Ecosystem Integration and Plug-ins

Dynatrace: The Dynatrace plug-ins page is displayed by categories, including ecosystem (Docker and SharePoint), Big Data (Splunk and Apache Cassandra), Notification (JIRA, PagerDuty) and many more. However, we're not sure if it's the lack of design or the relatively short list, but the library feels pretty basic to us.

AppDynamics: The company has an open platform for developers, allowing them to add their own plug-ins to the Exchange marketplace. The marketplace currently contains 140 plug-ins, and you can even send requests for the plug-ins you need and can't find.

Bottom line: We are always in favor of community based plug-ins that keep on growing thanks to the community.

5. Pricing

Both companies offer a valuation period to try out the pro version, followed by a downgrade to the free-and-lite version. If you want to know how much the pro version will actually cost, you'll have to work hard in order to find it out.

Dynatrace: Price is influenced by a number of factors, such as number of application servers, web servers, user sessions and other factors. You will have to talk with the sales team in order to get a proper quote, as pricing is not available on the Dynatrace website. The average deal size from other sources states is $10,000 for a yearly subscription.

AppDynamics: The company's free plan offers limited agent units per product module and limited data retention. Pro pricing is published on the AppDynamics website and starts at $3,600 per unit per year (e.g. JVM/OS Instance/Processes). If you're not from the US, a different pricing page loads up without the exact pricing details.

Bottom line: Both companies offer a free plan, but both differ when it comes to pricing transparency. AppDynamics publishes pricing, Dynatrace does not. However, both vendors have multiple products which are priced using different metrics so it's inherently complex to price for both unless you speak to someone in sales.

Who Should You Choose?

Both tools have been around for quite some time, although it feels like AppDynamics are the more relevant, up-to-date choice while Dynatrace is the old and "legacy" option. But we know that we can't judge an APM tool by its cover (or website, in this case).

Each company offers a wide range of analytics and analysis tools, so that you'll be able to understand what your users are experiencing and of course, monitor your systems. So if you're looking for a cut-through definitive decision, you're out of luck.

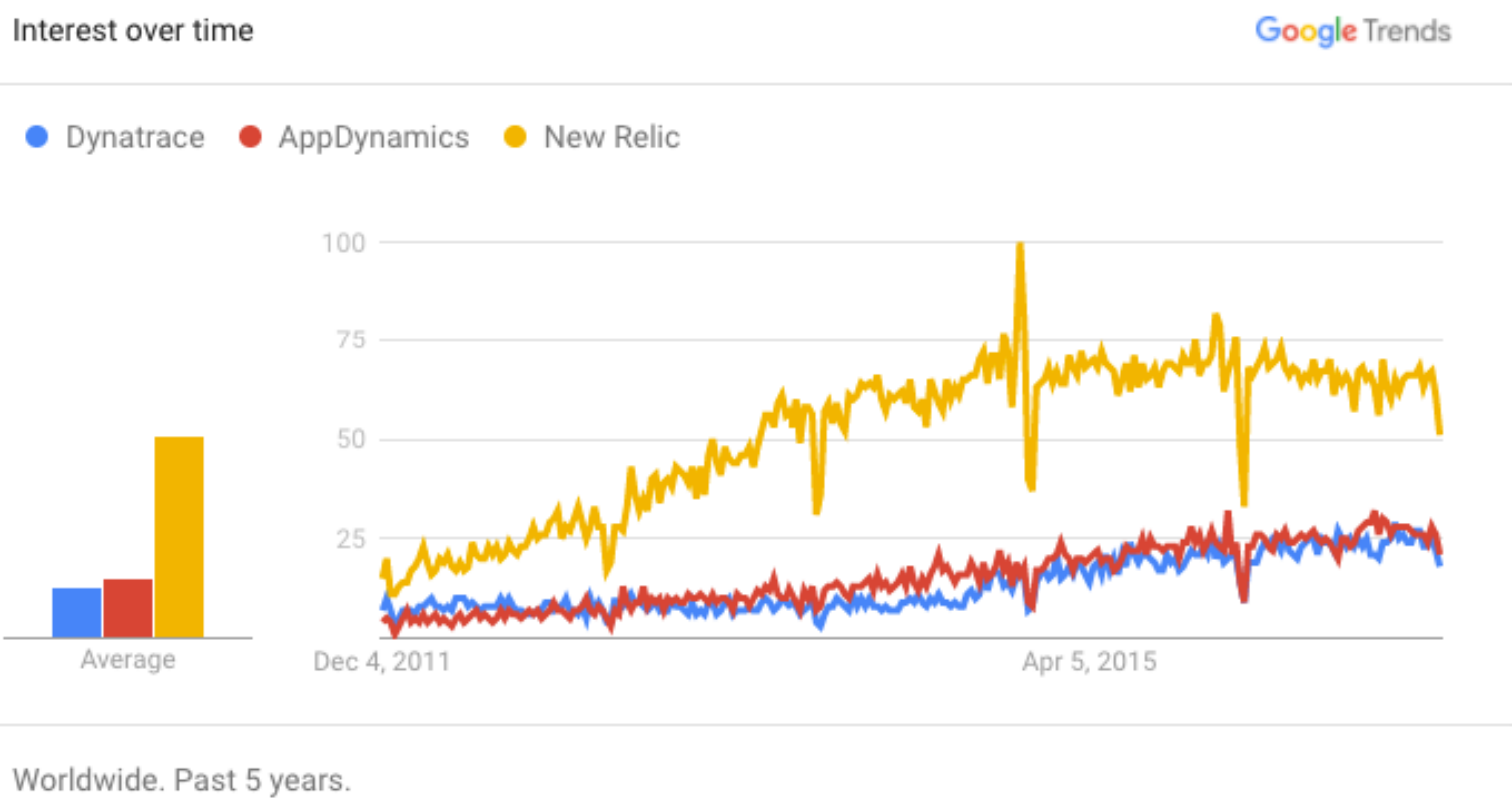
AppDynamics and Dynatrace offer a variety of features, some that might be ideal for you and some that might seem redundant. We encourage you to use the free trial with both companies in order to get a better sense of what you'll be dealing with and see which tool works better with your stack.

Chapter 2

Everyone's Talking About New Relic

When thinking about performance, New Relic is one of the main modern tools that come to mind. Spawned from the same company as AppDynamics, Wily Technology, who also dealt with performance monitoring and was acquired by CA back in 2006 – making way to new technology. New Relic is an anagram of Lew Cirne, its founder and CEO.

When it comes to popularity, there's no doubt that New Relic beats AppDynamics and Dynatrace, as can be seen on Google Trends:



But what does this interest meter means, and how does it affect your choice of an APM? That's why we've mapped our the features and options New Relic has to offer, so you'll be able to see how it fits your monitoring needs.

1. Supported Environments

Java, Scala, .NET, PHP, Node.js, Ruby and Python. Supported databases, cloud platforms and other plug-ins are available here. We'll dig in deeper with extensions later on.

On the user monitoring front, iOS, Android, and JavaScript support is included as well.

2. Features

New Relic can be broken down into 6 different products, all reporting to a main dashboard interface. Let's split these to backend, mobile and frontend to do a quick run-through over the main offerings.

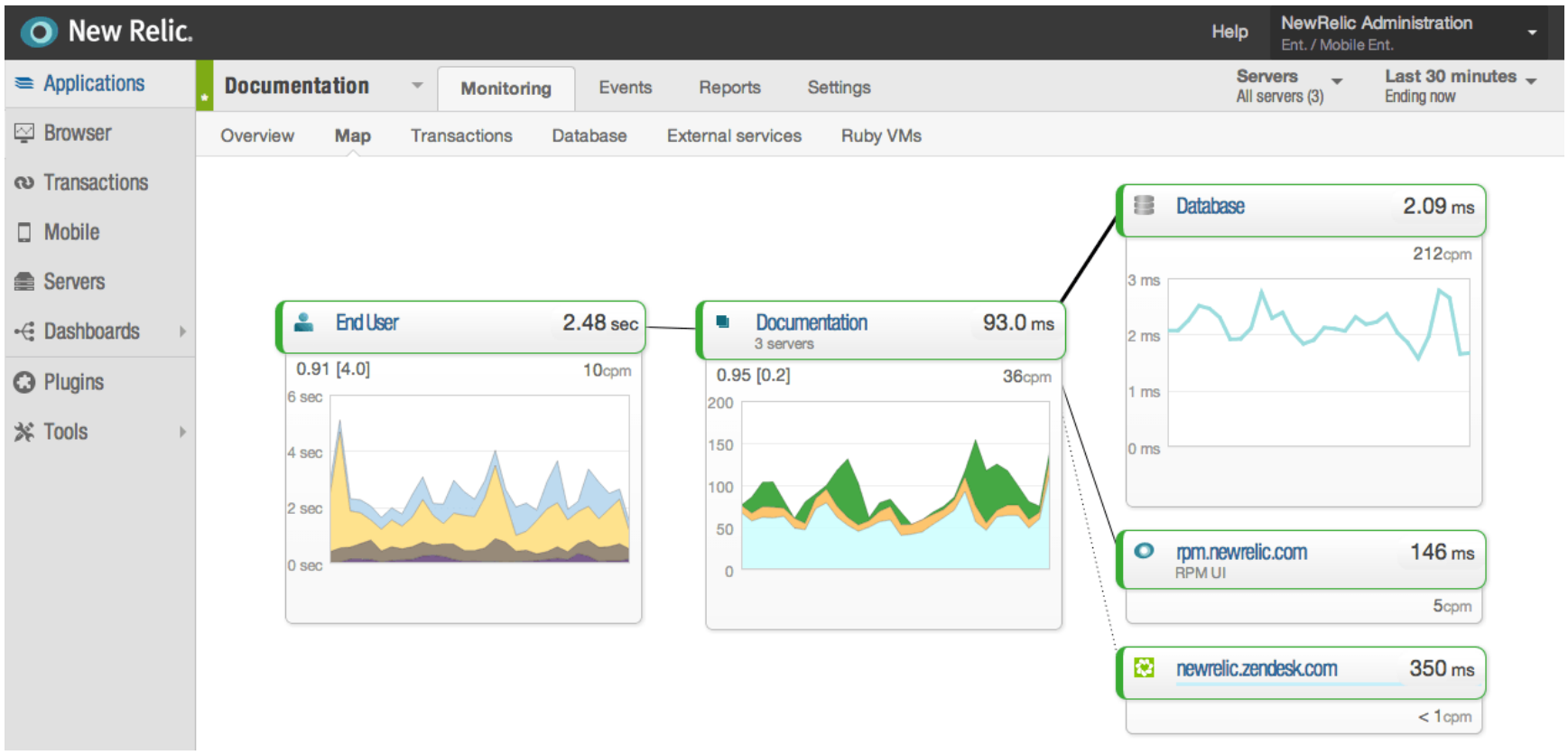
Backend Monitoring

The bread and butter of performance management – reporting stats, graphs and insights of your applications performance under the hood.

NewRelic offer 4 approaches here:

Application Performance Management

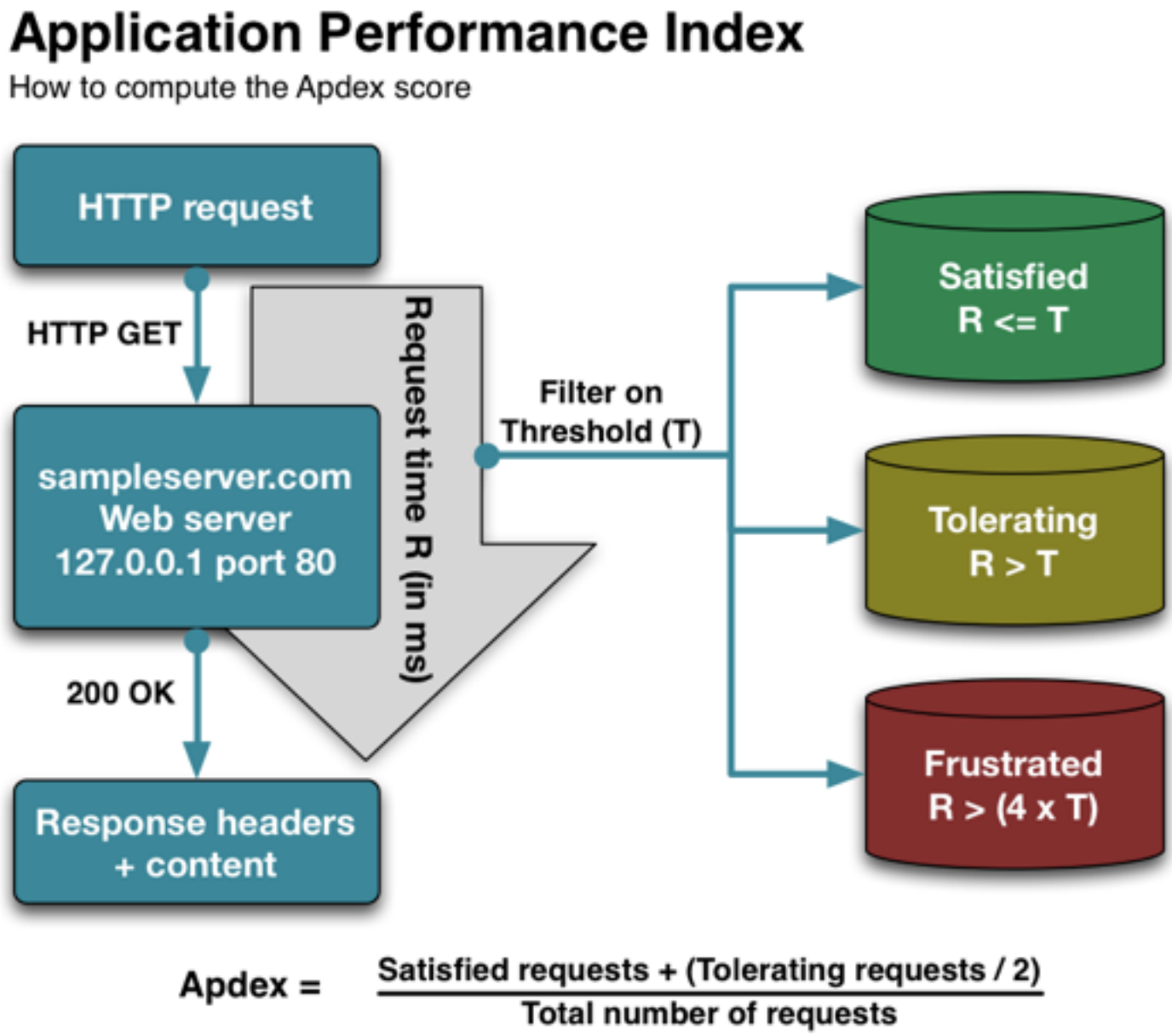
High level metrics with drill downs to code level data about how your application is performing. Must have metrics include transaction response time, error rate and throughput (Requests per Minute).



New Relic's application map

One of the thorny issues here is alerting and reporting, with so many metrics and moving parts, it's hard to identify which matters most. Is it a low error rate? Responsiveness? Throughput? New Relic is using the Apdex score index, which uses a user defined response time threshold T to imply end-user satisfaction. Simply put, they require you to manually set the threshold.

Here's an example for the way this score is calculated:



Calculating Apdex, now sum this over all requests for a given time and you'll get the score

Server Monitoring

Another monitoring capability offered by New Relic focuses on the hardware your servers run on: specs, CPU usage, memory utilization, disk I/O and network IO.

Database Monitoring

Moving on to other components in your stack, the first thing that comes to mind is the database. In New Relic, the Database dashboard is a part of the basic APM product.

It offers specific database monitoring metrics available through plugins to view data from external services (we'll talk a bit more about integrations later). Either way, both

native and external feature sets here might be different depending on the database you use.

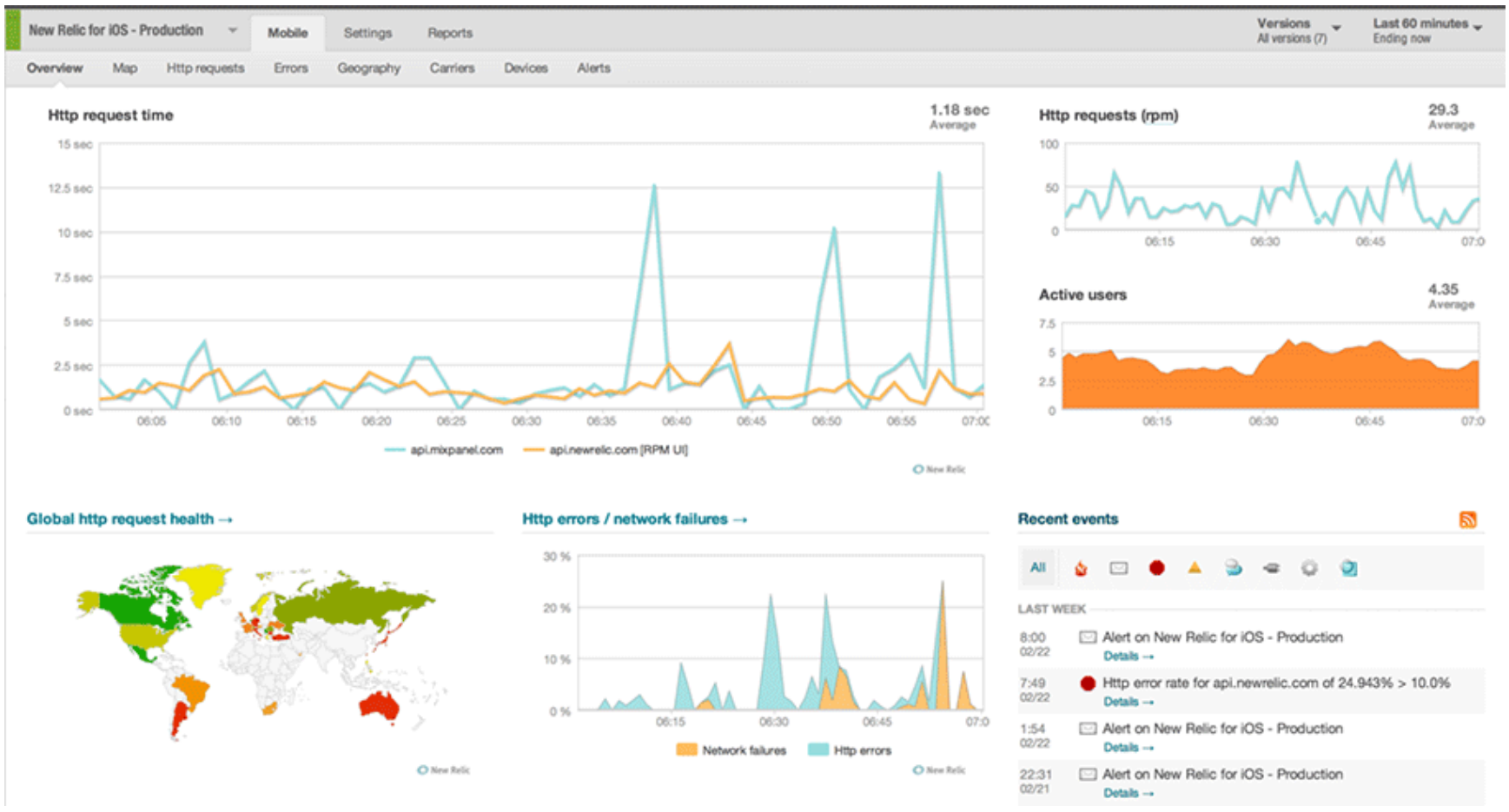
Insights and Analytics

This one is a wildcard, going beyond traditional APM and opening up to business intelligence metrics. Since New Relic already have access to the messages that go through your application, they've built this opt-in additional database to store your stats and enable you to query them.

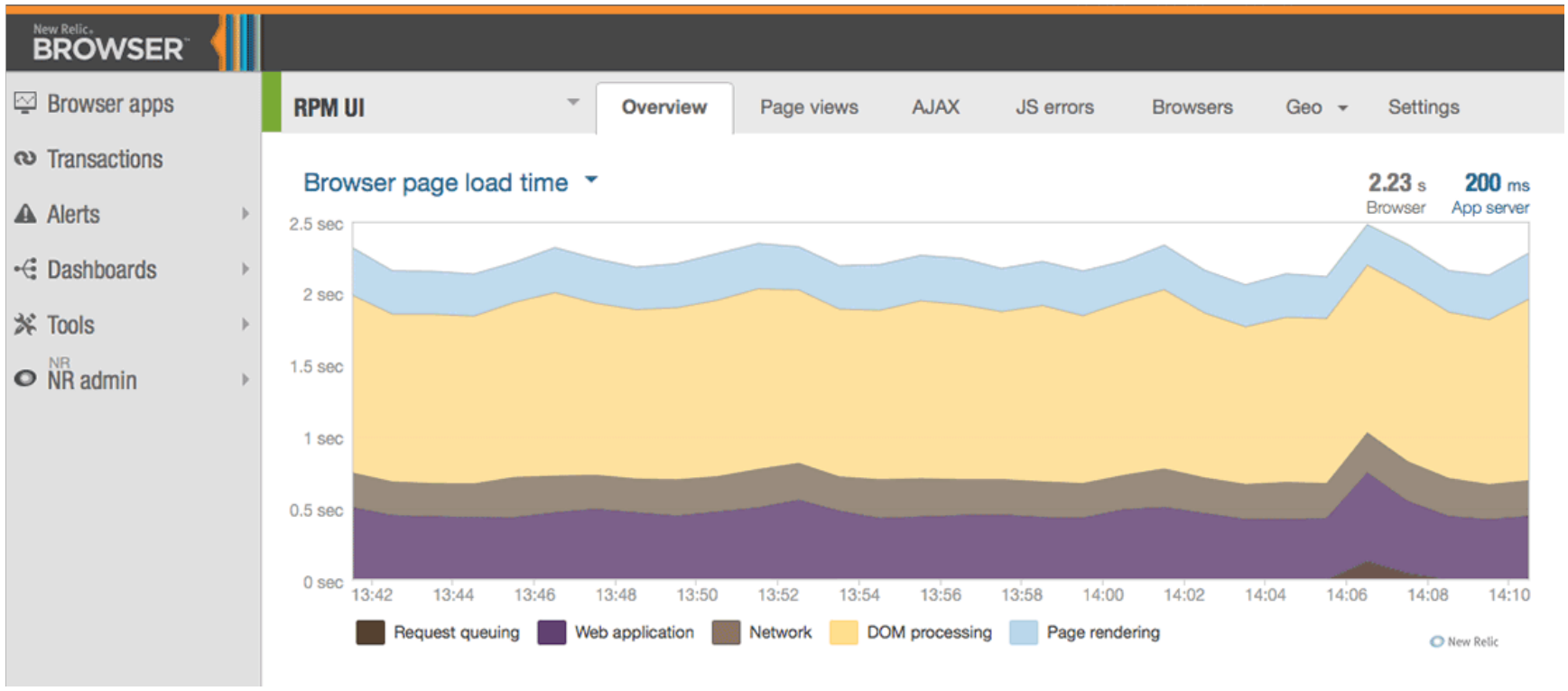
Frontend & Mobile Monitoring

Switching seats from the backend, lets take a quick look at what we're getting on the Real-User Monitoring front. New Relic have a product targeting browsers and a product targeting mobile with iOS & Android support.

On Mobile, the flagship features include insights on slowdowns and crashes, that are filtered through geographic regions, devices, operating systems and operator networks.



With end-user browser analytics, it feels like having the visibility you have on your browsers load times through Chrome dev tools available on the actual users of your app.



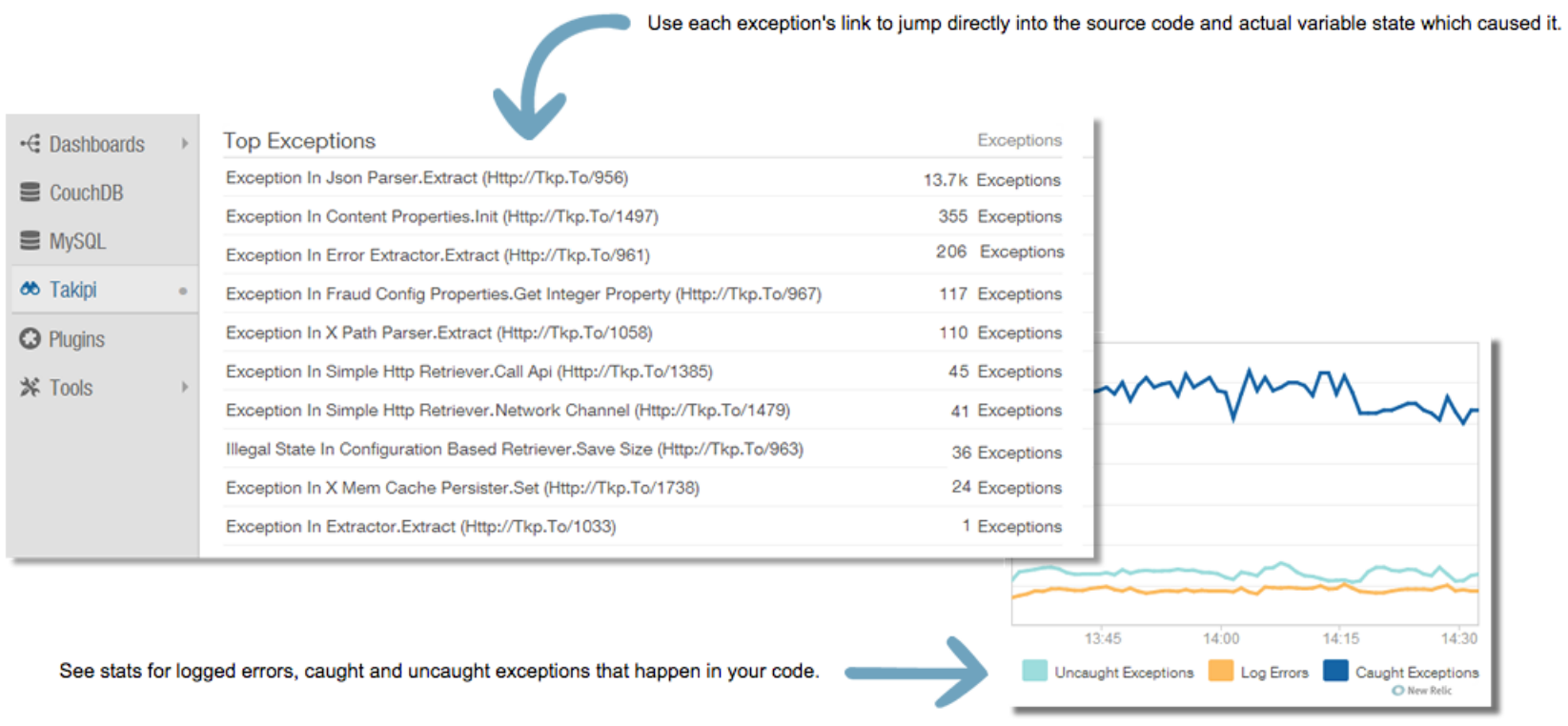
3. How to Solve the errors you find

To go beyond the reporting and alerting of errors by New Relic, many of our users add OverOps to their toolbox. This allows them not only to monitor server slowdowns and errors via New Relic, but also to solve them using OverOps.

Whenever a new exception is thrown or a log error occurs – OverOps captures it and shows you the variable state which caused it, across methods and machines. OverOps will overlay this over the actual code which executed at the moment of error – so you can analyze the exception as if you were there when it happened.

The dashboard links each error to a recorded instance of all involved code when the bug happened, and includes the variable values that caused it.

OverOps also has a New Relic plug-in that displays an exception and log error dashboard:



OverOps for New Relic

It's one thing to identify what's causing troubles in your application, but solving it, is a whole different issue. Java or Scala developers? Whether you're using an APM tool or not give OverOps a try.

4. Dashboard and Usage

To get a better feel of each tool's user experience and way of solving problems, I think it's probably best to browse through a video. NewRelic at TypeSafe, a webinar that gives an overview of New Relic for Play (it's a bit long but gives a nice overview if you browse through):

https://www.youtube.com/watch?v=LJL7EAavb1Q

5. Installation

New Relic is only available through SaaS.

Agents: Monitoring your application becomes available through attaching language specific agents to your server. For example, with Java there are 2 possible ways to

instrument your code with agents, either by using a Java agent or a native agent. New Relic use a Java agent to collect the performance data they're reporting. To gather the low-level data required not only to point to an error but to help solve it, OverOps uses a native agent.

Code and configuration changes: On the Real-User Monitoring front, project and configurations changes including introducing a few dependencies would be needed if you'd like to add monitoring capabilities to your web or mobile app. This includes adding JS agents to your website and native mobile agents to your mobile application.

Alerting: New Relic relies on custom thresholds defined by you for its Apdex index.

6. Integrations and Plugins

Branching out, New Relic offer integrations and plug-ins to hundreds of services. We've already mentioned the Platform program earlier: a plug in platform with 116 (Last time I checked) plugins to services like Hadoop, RabbitMQ and Redis, that stream metrics of their data so you can view in on New Relic. On the integrations side of the table, there's Connect, with 53 integrations with tools like Jira, HipChat, OverOps and pagerduty.

7. Pricing

There's a free lite version with limited features across all products, including a 24hr data retention with pro trials of 14-30 days.

Pro account pricing starts with with $199 per month per host ($149 on a yearly plan), this includes APM, Servers, Platfrom and Browser basics. Mobile monitoring costs $49 per month ($29 on a yearly plan) with 1 week of data retention. The Insights product start from $250 per month for up to 75 million events.

New Relic is top of the line APM tool, that can fit different type of developer, from enterprises to startups. The choice is not clear, but you could not go wrong – If you're looking for an on premise solution, AppDynamics or Dynatrace may be the better options. Otherwise, it's an individual call depends on which better fits your stack (and which of all these features are you actually thinking you're going to use).

Chapter 3

5 Open Source Tools You Should Know

Little known yet useful: The state of open source Application Performance Monitoring

One of the most important things for any application is performance. We want to make sure the users are getting the best experience they can, and to know that our app is up and running. That's why most of us use at least one monitoring tool.

If you're looking for something a little different in the performance monitoring market, one option you can choose is going for an open sourced tool. In the following post we've gathered some open source APM tools that are available today as an alternative to the paid tools, so you'll be able to see if it's the right choice for you.

Going Open Source

The APM market is a crowded one. Since there are so many players in the game, and they all know the value of monitoring your application, they keep their code for themselves.

However, there's an alternative: open source tools. These tools present a good option if you're interested in an easy way to gain visibility for you application in production, and if you want to know how your code is actually being monitored.

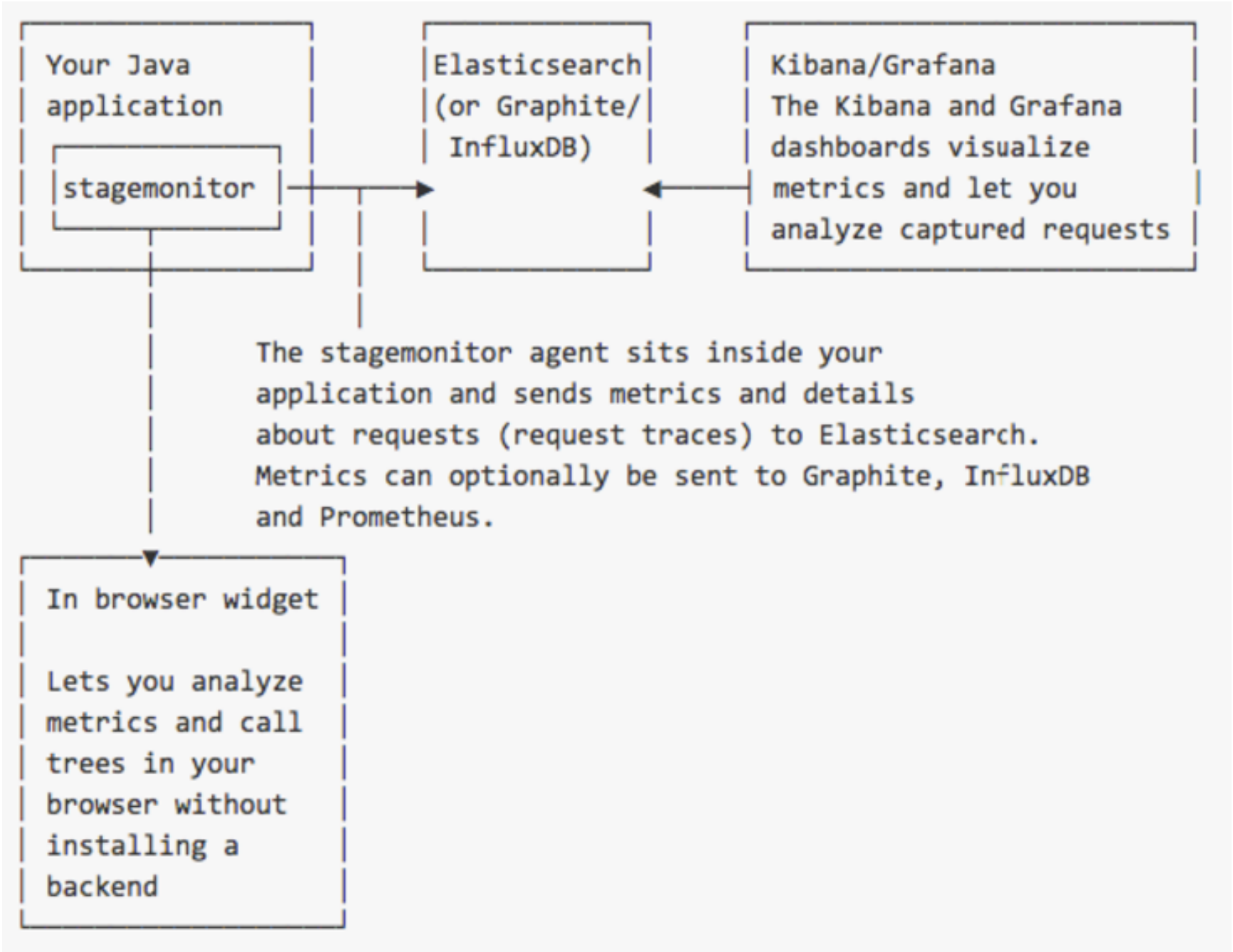
There are a few key APM tools in the open source community as well, each with its own offerings and possibilities. If you're intrigued and want to know exactly what open source APM has in store for you, we've covered the top 5 tools available for you:

- Stagemonitor
- Pinpoint
- MoSKito
- Glowroot
- Kamon

1. Stagemonitor

Stagemonitor offers a Java monitoring agent, that was built with clustered application stacks in mind. Meaning that it aims to monitor applications that are running on a number of servers. The tool integrates with time series databases (TSDB). This tool is optimized for handling time series data, along with arrays of numbers that are indexed by time. These databases include Elasticsearch, Graphite and InfluxDB.

Architecture



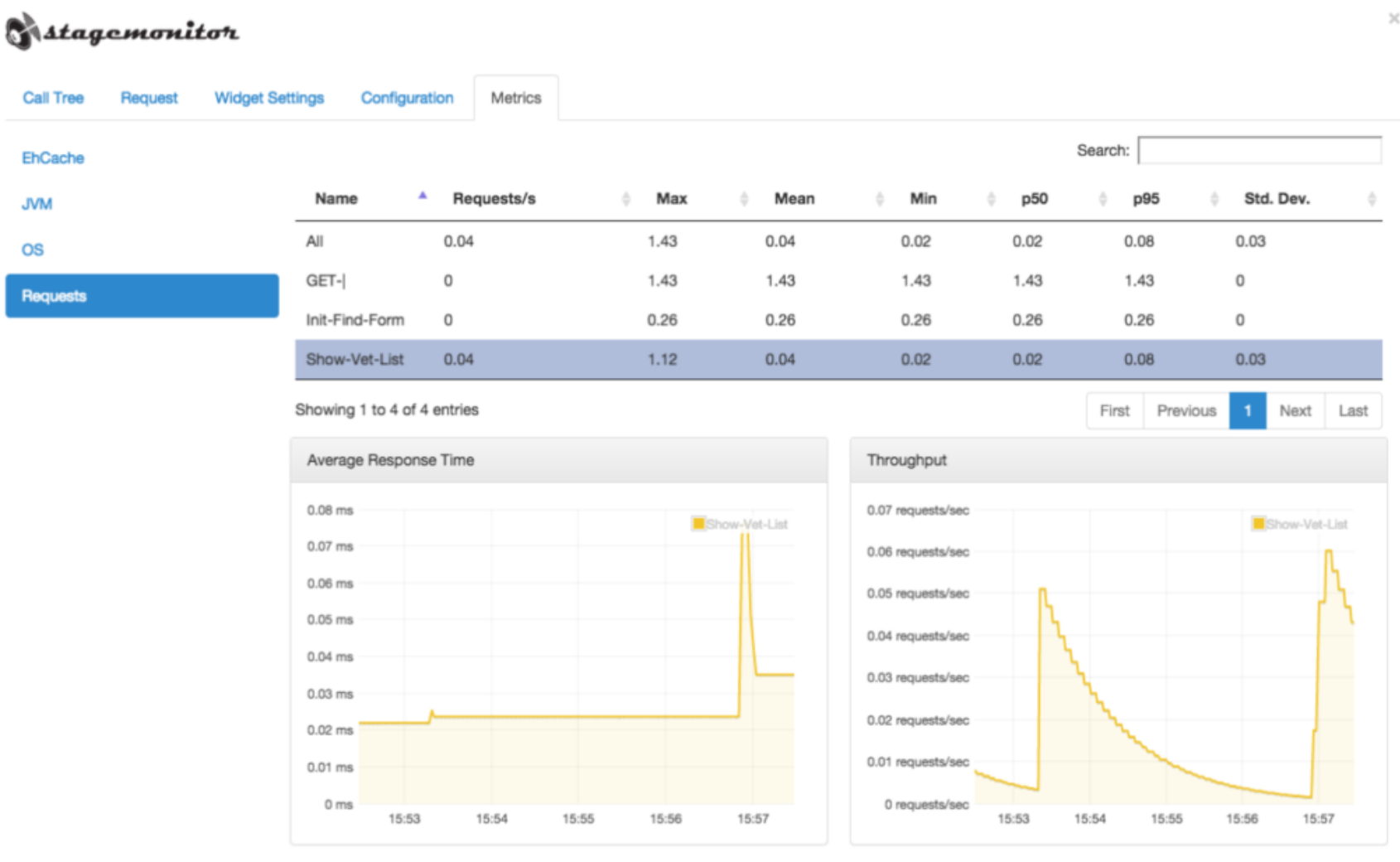
How does it work?

Stagemonitor includes an agent that sits in your Java application, sending metrics and request traces to the central database. The tool only requires one instance to monitor all applications, instances and hosts and can be deployed inside your own datacenter.

On the monitoring side, you can view historical or live data from the cluster, or directly from the developer server, create custom alerts and define thresholds for each metric.

Stagemonitor includes a dashboard, so you can visualize and analyse the different metrics and requests you're interested in. You can create custom dashboards, write your custom plugins or even use 3rd party plugins. It offers an in-browser widget with no backend required that is automatically injected to the monitored webpage. You can view the live demo in the following link.

In the official documentation, Stagemonitor states that it offers support for non servlet based applications as well, and you can check out the full requirements here.



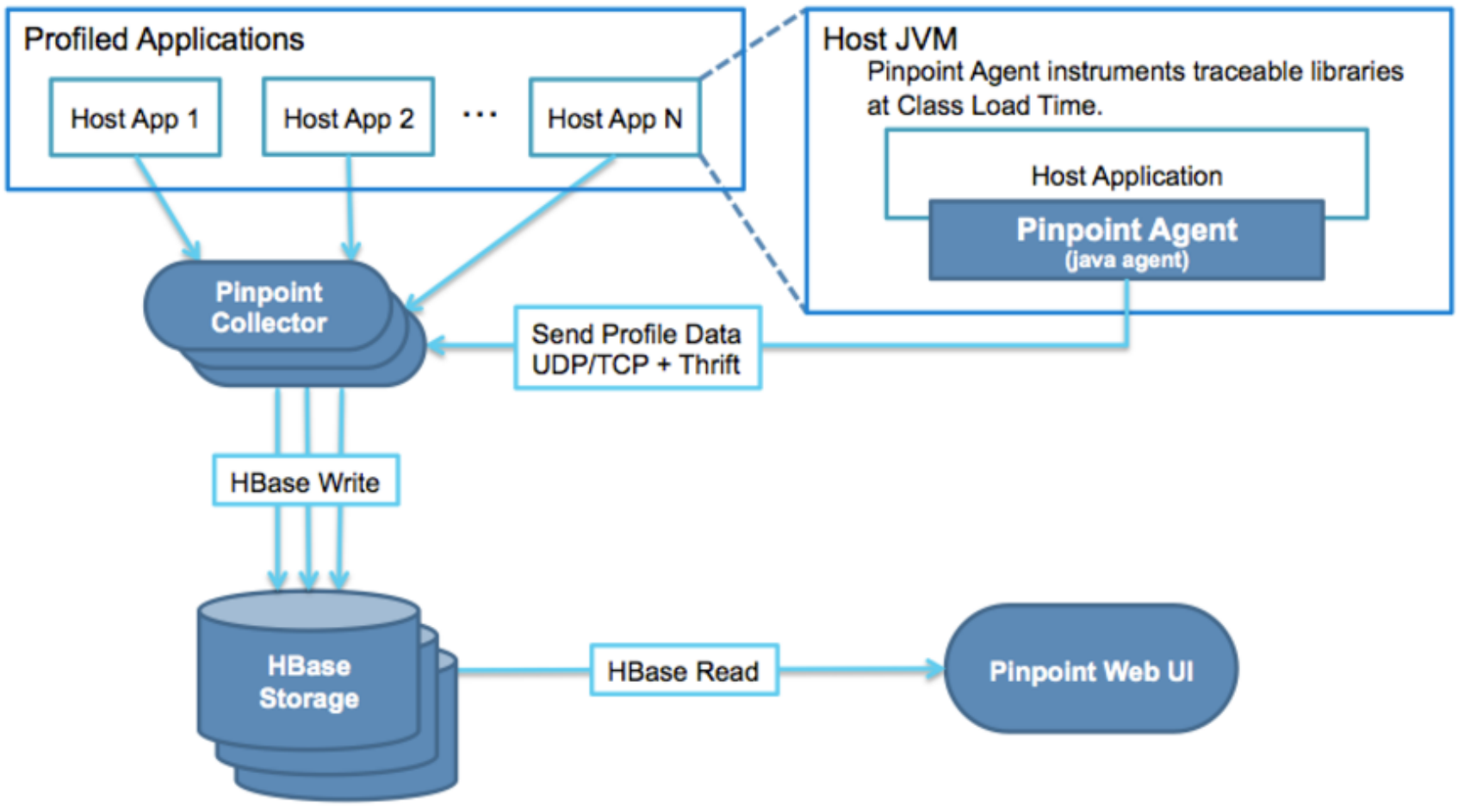
Stagemonitor's Widget Metrics

Bottom line: If you're already familiar with the ELK stack, it's definitely worth checking out for a quick test run.

2. Pinpoint

Pinpoint is an APM tool made for large scale distributed systems. It's modeled after Dapper, a distributed systems tracing infrastructure built by Google, providing its developers more information about the behavior of complex distributed systems.

Architecture

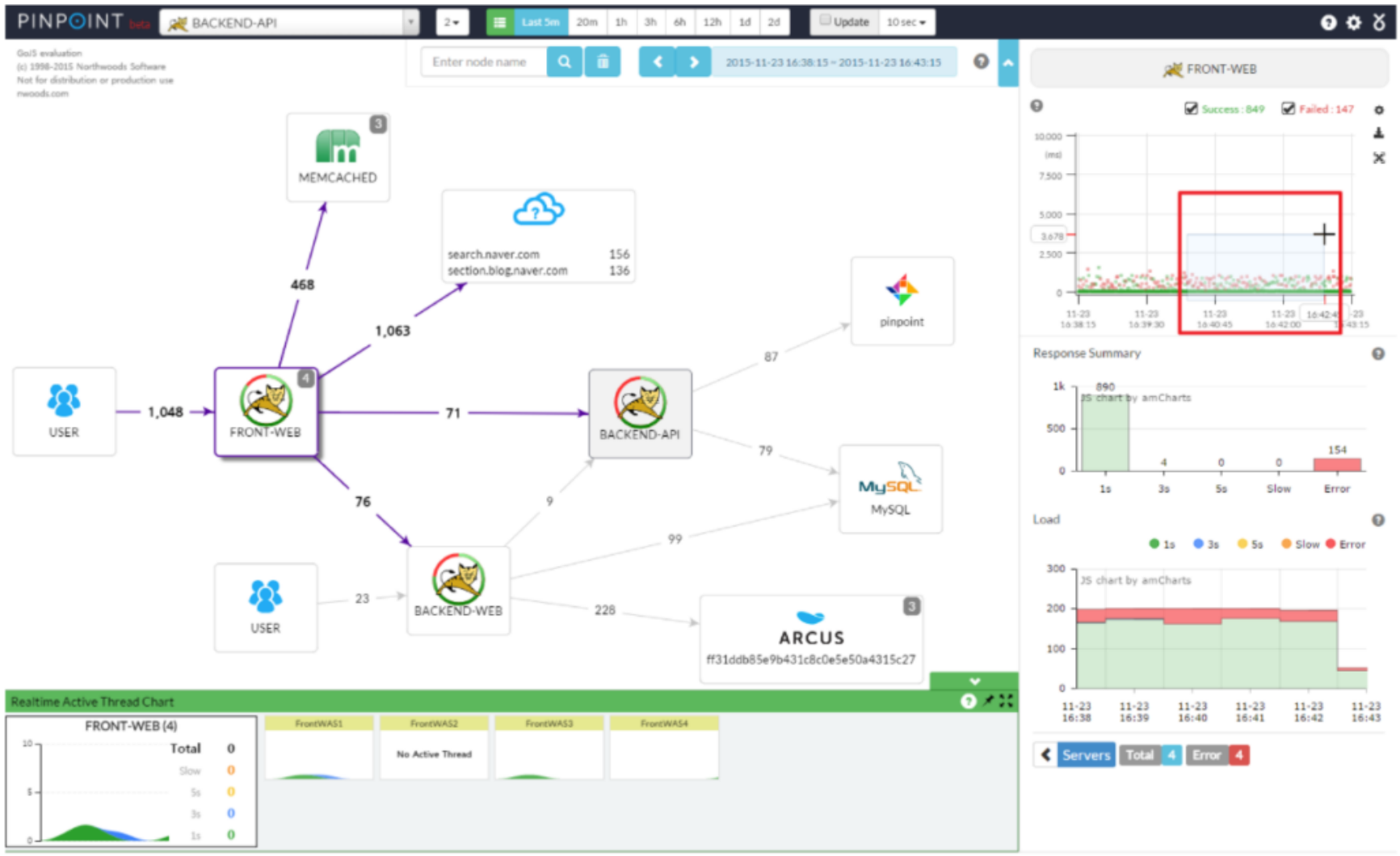


How does it work?

The tool helps analyze the overall structure of the system and how components within them are interconnected, by tracing transactions across distributed applications. Meaning that it aims to explain how every transaction gets executed, trace the flows between the components and (bad joke ahead) pinpoints problematic areas and potential bottlenecks.

The dashboard helps visualize how the components are connected, and lets you monitor active threads inside the applications in real time. Pinpoint also lets you see the request count and response patterns so you'll be able to identify potential problems. You can view critical details that include CPU usage, memory/garbage collection and JVM arguments.

Pinpoint works with an agent that's installed without any code changes, and you can run a sample instance in your own machine by running four simple scripts for each of the components: Collector, Web, Sample TestApp and HBase.



Pinpoint's ServerMap

Bottom line: If you've heard of Dapper, or would like to monitor and analyze your complex distributed systems, you should definitely check this tool out.

3. MoSKito

MoSKito offers 3 tools in one:

MoSKito-Essential – The basic standalone project. It's the core of MoSKito functionality that lets you monitor your application

MoSKito-Central – Centralised storage server for keeping the performance data

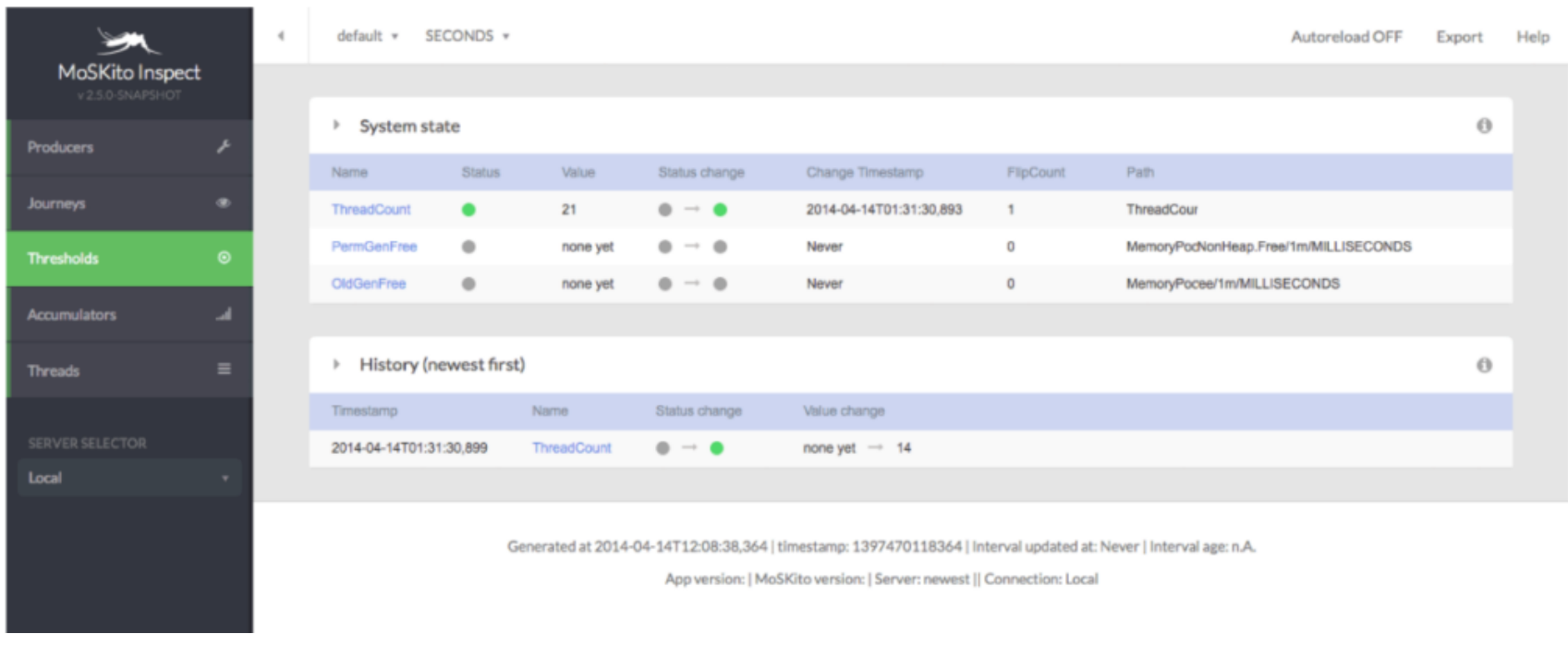
MoSKito-Control – A tool for monitoring performance of multi-node web applications

How does it work?

To get started, all you need to do is drop the .jar file into the WEB-INF/lib folder or by including a small new section in the web.xml file. Once the tool is up and running, it collects performance data, analysing it in real time as well as storing it for historical analysis.

The tool collects all of your performance metrics, such as threads, memory, caches, storage, services, registrations, payments, conversion, SQL, load distribution and so on. It doesn't require code change, supports all of the major app servers (Tomcat, Jetty, JBoss, WebLogic) and keeps the data locally.

You also get a notification system to know when a threshold was met, and the recordings of user's actions you want to monitor. Along with the web-based dashboards, the tool also offers a mobile app to monitor your application on the go.



MoSKito's Essential

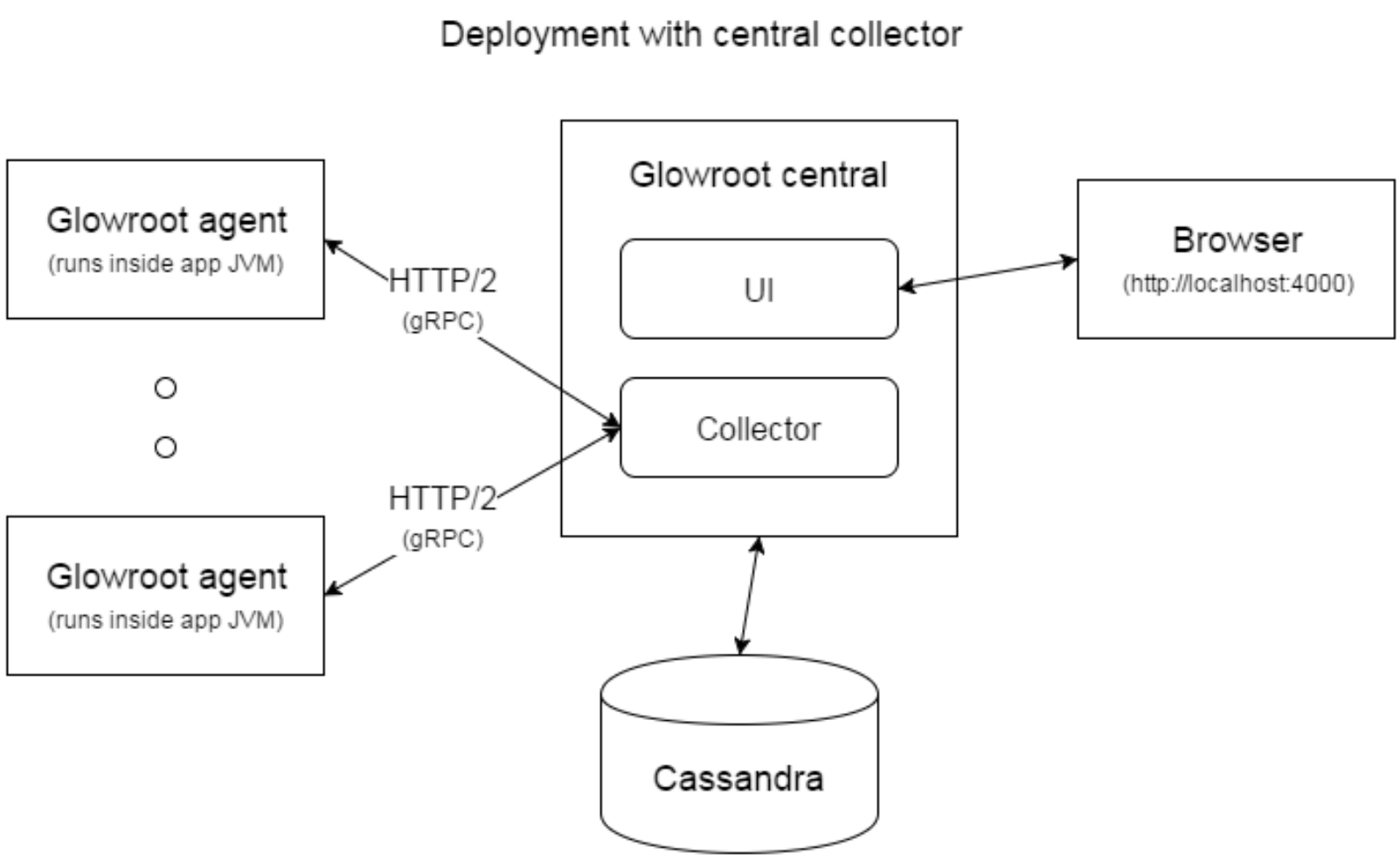
Bottom line: MoSKito first launched at 2007, and by now it's a well known and stable tool, that's supported by the team and by the community, including paid support options. That's also a huge plus for any open source tool.

4. Glowroot

Glowroot prides itself on being a fast, clean and simple APM tool. It will allow tracing capture for slow requests and errors, and you'll be able to log time trace for each user action, as well as SQL capture and aggregation. The tool also presents a historical rollup of all data with configurable retention.

It provides charts to visualize response time breakdown and response time percentiles, and its responsive UI will allow you to monitor your application from your mobile devices as well as your desktop.

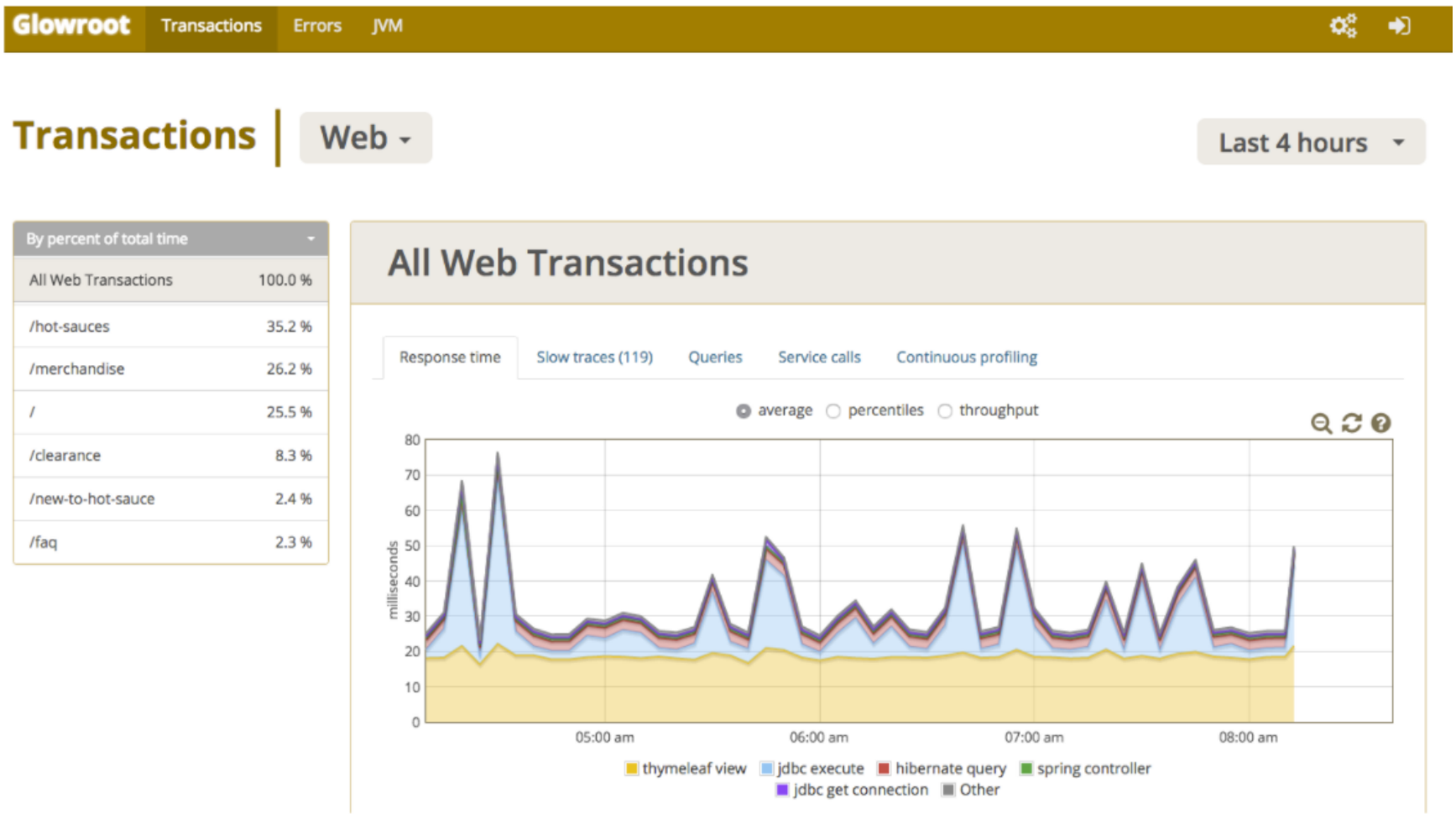
Architecture



How does it work?

To get started with Glowroot, you need to download and unzip the main installation file and add -javaagent:path/to/glowroot.jar to your application's JVM arguments. After you start your application, all that's left is pointing the browser to http://localhost:4000.

Once the tool is up and running, you'll get continuous profiling (with filtering options), along with being able to set up alerts for response time percentiles and MBean attributes. Glowroot offers full support for async requests that span multiple threads, and it supports Tomcat, TomEE, JBoss EAP, Wildfly, Jetty and Glassfish.



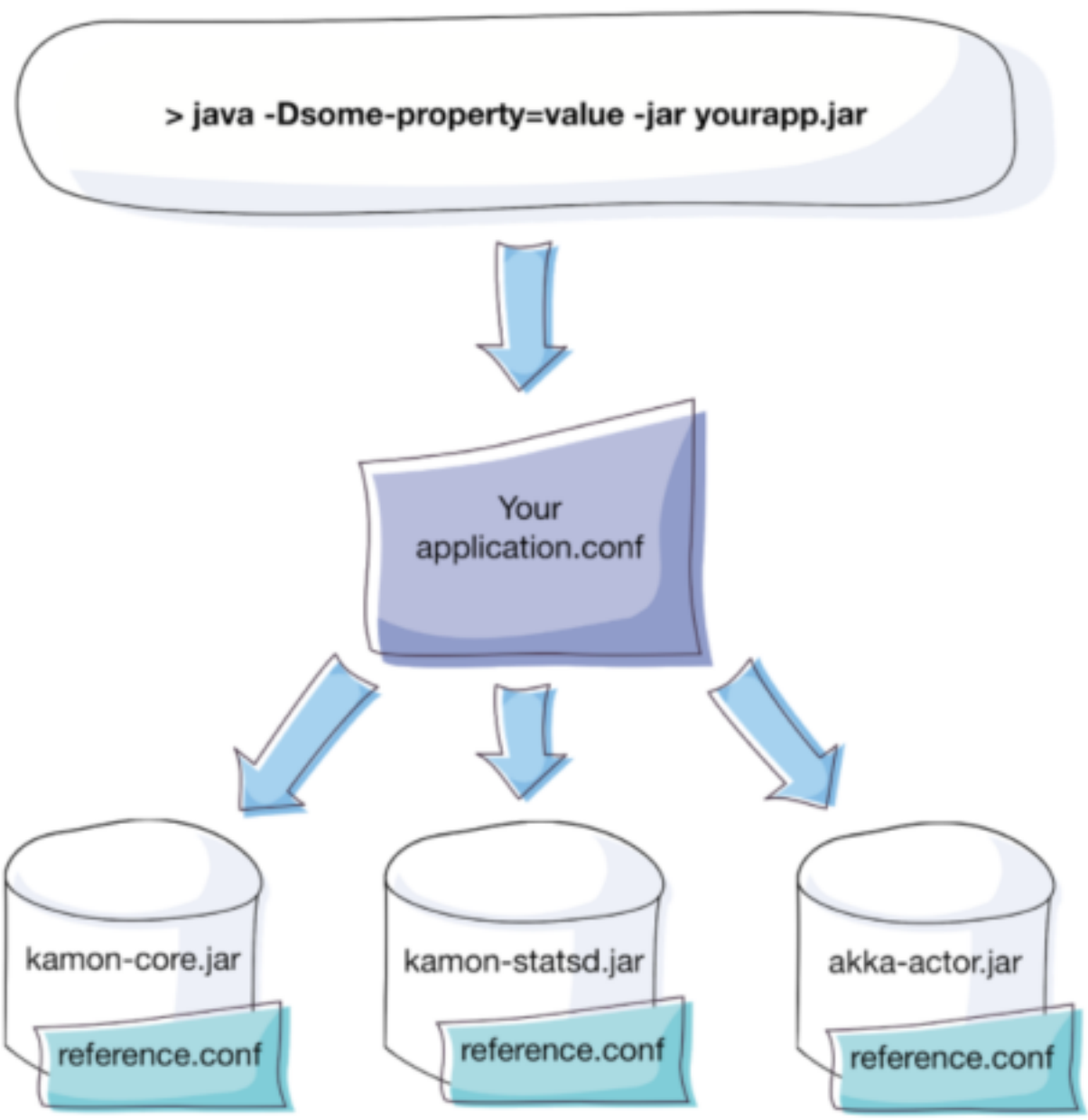
Glowroot's Dashboard

Bottom line: If clean and simple is what you're looking for, no doubt you'd want to check out Glowroot over the other tools here.

5. Kamon

Kamon is a reactive-friendly toolkit that is built for applications that run on top of the JVM. More specifically, it's made for applications that are built with the Typesafe Reactive Platform (using Scala, Akka, Spray and/or Play!), but still offers support for any other JVM platforms and languages.

Architecture



How does it work?

Kamon is distributed as a core module with all the metric recording and trace manipulation APIs and optional modules that provide bytecode instrumentation and/or reporting capabilities to your application. Or in other words, it offers a simple API for recording metrics and trace information for JVM applications.

All of Kamon's modules are available through Maven Central, and you will need to add them as a compile dependency to your project. Once you've included the modules you're interested in, simply start up Kamon, and all of the available modules will be automatically started, you don't need to explicitly activate/start them.

The tracing modules will allow recording data about functionality executed in your application, and the metrics module will allow you to control the registration of entities being tracked either by user code or by instrumentation provided with other Kamon modules. It also has other abilities such as filtering, configuring instrument factories and dispatching metrics subscriptions.

Bottom line: If you're using a number of JVM languages, or mostly Scala / Akka, and would like "one tool to monitor them all", Kamon might be the friendliest choice to go for.

Now that you've got your haystack…

APM tools are great at giving you the information about whether your application is up and running, or if there's something that's holding it back. The only problem is that once you get that haystack in which the problem was found, you have to start digging around looking for the actual needle that caused it.

Instead of sifting through log files trying to locate what went wrong, where it happened and what might have caused it – there's a better solution. OverOps will not only give you the answers to where and when, it will also show you why the error happened – giving you the complete source code and variable state that caused an error, across the entire call stack. Check it out.

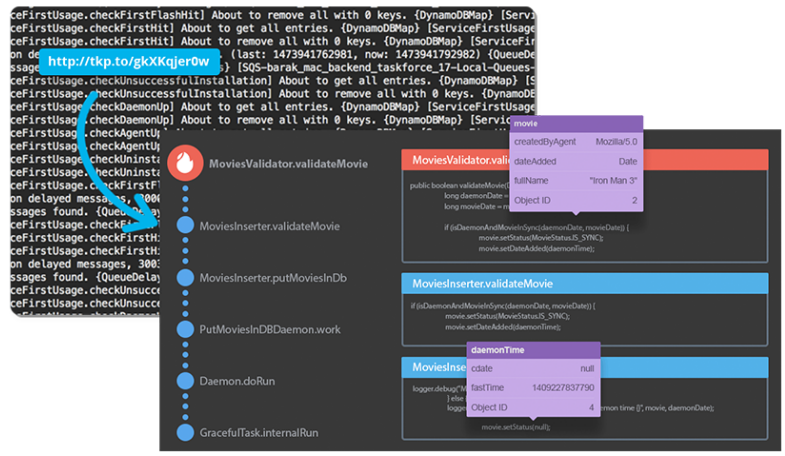
Final thoughts

These are some good alternatives to the paid tools in the APM space. BUT… Some might think that going for the open source option is mostly a way to save a few bucks. It's also important to remember that while you won't need to issue an invoice for the use of the tool, it doesn't necessarily mean it's cheaper.

Open source tools come with a price: installation, troubleshooting and of course maintenance, which will all be done in house, by your very own engineers or even you. And not to mention the time you might end up wasting seeking support for a specific issue only you've encountered, and the community never heard of.

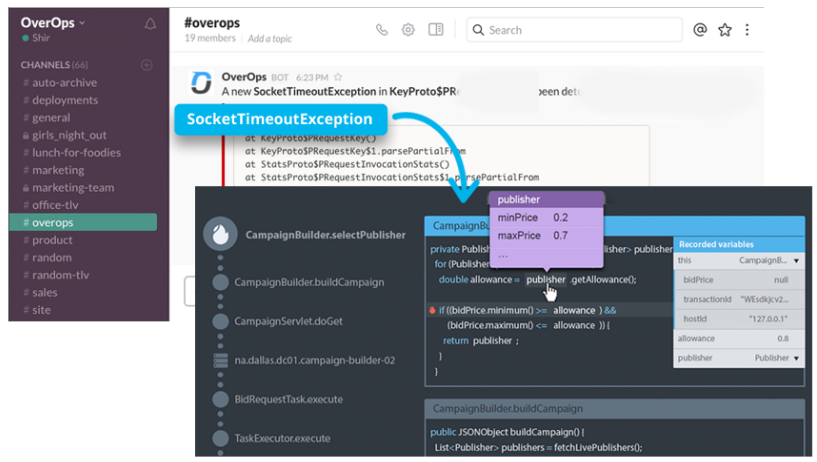
Our 2 cents is that open source can be great, but you should also keep the other costs in mind and only then reach a decision.

Jump to the source code and variables behind any error on your logs.



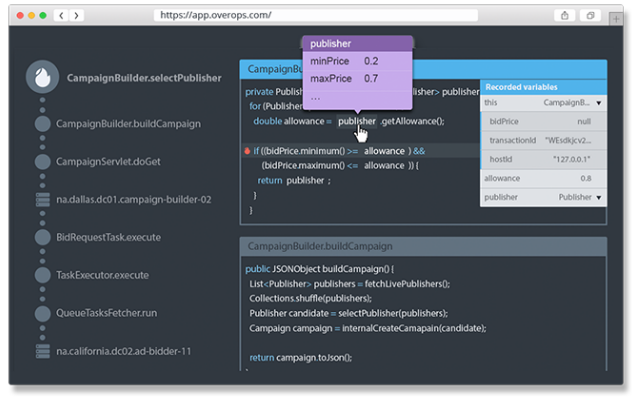
splunk> elastic sumologic loggly

Get real-time push notifications for critical errors through the tools already use.



JIRA slack HipChat pagerduty

See the source code & variables across the entire call stack behind any production error.



SaaS, Hybrid & On-premises

Final Thoughts

Monitoring tools are an essential inclusion in production environments today. Visualizing metrics, tracking errors, monitoring performance, and analyzing your application are all key activities for gaining insight into the workings of your application. Recognizing the need is easy, but choosing which monitoring tool or set of tools to use can be difficult.

Take advantage of the free trials the big tool have to offer, and make some time to check out the open source options as well. The bottom line is that the best way to see if something fits you, is to try it out for yourself.

We hope you've found this guide useful and would be happy to hear your feedback on twitter @overops and over email: hello@overops.com