APPDYNAMICS

An AppDynamics Business White Paper

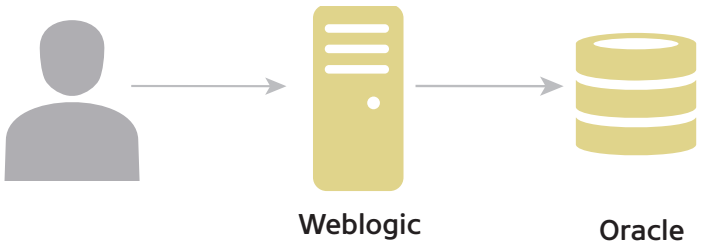
WHEN LOGGING ISN'T ENOUGH

A Modern Approach to Monitoring Performance in Production

Ten years ago, the standard way to troubleshoot an application issue was to look at the logs. Users would complain about a problem, you'd go to operations and ask for a thread dump, and then you'd spend some time poring over log files looking for errors, exceptions, or anything that might indicate a problem. There are some people who still use this approach today with some success, either because their applications are very simple or because their technology is not yet supported by most monitoring tools, but for most modern applications logging is simply not enough anymore. If you're depending on log files to find and troubleshoot performance problems, then chances are your users are suffering – and you're losing money for your business. In this white paper we'll look at how and when logging is no longer enough for managing application performance.

The Legacy Approach

Ten years ago, logging was the best (and only) way to investigate performance issues. The typical legacy web application was monolithic and fairly static, with a single application tier talking to a single database that was updated every six months. The legacy approach to monitoring production web applications was essentially a customer support loop. A customer would contact the support team to report an outage or bug, the customer support team reports the incident to the operations team, and then the operations team would investigate by looking at the logs with whatever useful information they had from the customer (username, timestamps, etc.). If the operations team was lucky and the application has ample logging, the operations team will spot the error and bring in developers to find the root cause and provide a resolution. This is the ideal scenario, but more often than not the logs were of very little use and the operations team would have to wait for another user to complain about a similar problem and kick off the process again. Ten years ago, this was what production monitoring looked like. Apart from some rudimentary server monitoring tools that could alert the operations team if a server was unavailable, it was the end users who were counted on to report problems.



Limitations of Logging

Trawling log files has never been particularly easy or fast. Even if you have a monolithic, unchanging application there are still problems with using logs to manage application performance, especially in production.

Logging is Inherently Reactive

The most important reason that logging was never a great strategy for managing performance is that logging is an inherently reactive approach to performance. Typically this means an end user is the one alerting you to a problem, which means that they were affected by the issue – and (therefore) so was your business. A reactive approach to application performance loses you money and damages your reputation.

You're Finding a Needle In a Haystack

Another reason why logging isn't suitable for production is that system logs have a particularly low signal to noise ratio. This means that most of the data you're looking at (which can amount to terabytes for some organizations) isn't helpful. Sifting through log files can be a very time-consuming process, especially as your application scales, and every minute you spend looking for a problem is time that your customers are being affected by a performance issue. Of course, newer tools like Splunk, Loggly, SumoLogic and others have made sorting through log files easier, but you're still looking for a needle in a haystack.

Logging Requires an Application Expert

Even with tools like Loggly and Splunk, you need to know exactly what to search for before you start, whether it's a specific string, a time range, or a particular file. This means the person searching needs to be someone who knows the application well, usually a developer or an architect. Even then, their hunches could be wrong, especially if it's a performance issue that you've never encountered before.

Not Everyone Has Access To Logs

Logging is a great tool for developers to debug their code on their laptops, but things get more complicated in production, especially if the application is dealing with sensitive data like credit card numbers. There are usually restrictions on the production system that prevent people like developers from accessing the production logs. In some organizations, these can be requested from the operations team, but this step can take a while. In a crisis, every second counts, and these costly processes (while important) can cost your organization money if your application is down.

You're Only Seeing Part of the Picture

Even in a perfect world where you have complete access to your application's log files, you still won't have complete visibility into what's going on in your application. The developer who wrote the code is ultimately the one who decides what gets logged, and the verbosity of those logs is often limited by performance constraints in production. So even if you do everything right there's still a chance you'll never find what you're looking for.

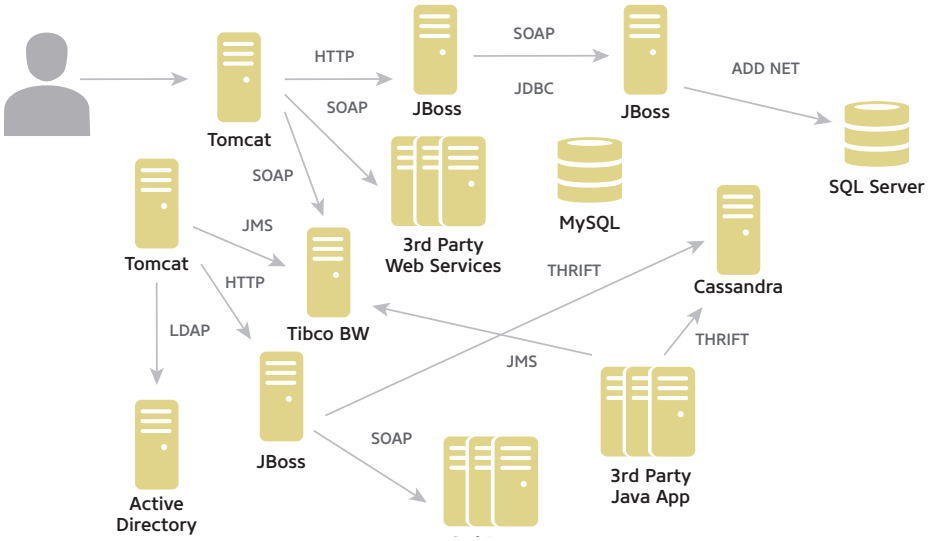
Furthermore, your logs can only tell you what's going on inside your application servers. What if the bottleneck is in a third party service call? Or the infrastructure? Logs can't tell you what's happening on the edges of your application, or on the machines themselves. When it comes to preventing downtime you need to monitor not only your application, but also your infrastructure and third party dependencies. Is the machine CPU maxed out, or does this error only happen when disk I/O is maxed out? Does the downtime only happen when a third-party web service is slow? Logging won't tell you about anything except what a developer thought would be useful at the time.

The New Normal – When Logging is No Longer Enough

Logging has never been perfect, but it was pretty widespread ten years ago, simply because there weren't many alternatives. Today, however, there are application performance management solutions for the all the most popular programming languages that make it much easier to troubleshoot performance bottlenecks. Unless you're using bleeding-edge technology that isn't supported by monitoring tools like these, you have no reason to rely on logging for managing performance. In fact, there are some very good reasons not to rely on logging anymore.

Your App is Much More Complex

Today, enterprise web applications are much more complex than they were ten years ago. The new normal for these applications includes multiple application tiers communicating via a service-oriented architecture (SOA) that interacts with several databases and third-party web services while processing items out of caches and queues. The modern application has multiple clients from browser-based desktops to native applications on mobile. As a result, it can be difficult just to know where to start if you're depending on log files for troubleshooting performance issues.



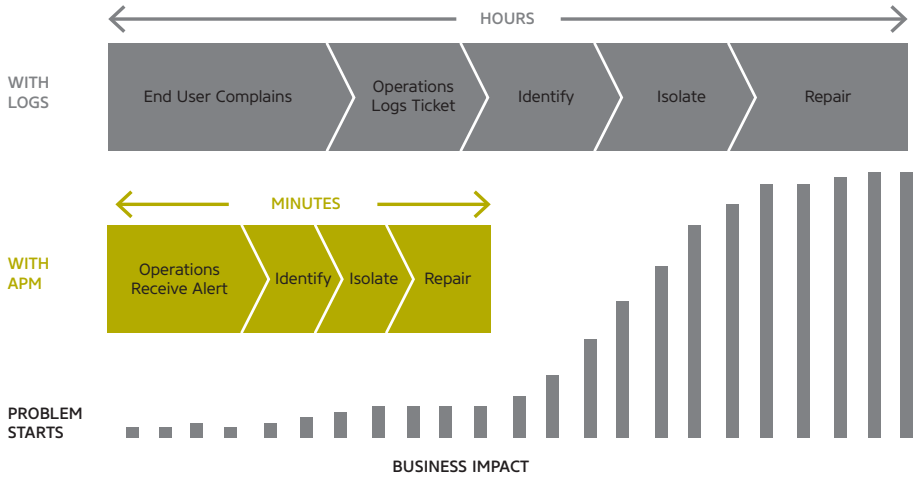
Your Business Runs On Apps

Every company is a software company now, and every company is invested in the performance of their applications. More and more businesses get their revenue through their website rather than their storefront, and they can't afford to lose money if their web applications are slow. And the definition of "slow" changes daily as your end users become accustomed to faster and faster speeds. Here are a few recent findings about web performance and revenue:

- Microsoft found that Bing searches that were 2 seconds slower resulted in a 4.3% drop in revenue per user.
- When Mozilla shaved 2.2 seconds off their landing page, Firefox downloads increased 15.4%.
- Shopzilla saw conversion rates increase by 7-12% as a result of their web performance optimization efforts.
- Making Barack Obama's website 60% faster increased donation conversions by 14%.

The reality is that your business is affected by performance issues long before your service goes down. The longer it takes to find the root cause of those performance problems, the more money you lose.

How much money are we talking about? One large e-commerce company in the United States decided to calculate the cost of a recent performance problem caused by code deadlock. They found that over 2,000 transactions were effected by the deadlock, and at an average of $74 per transaction, they found they had lost over $180,000 in business. Not every company is the same, but if your business relies on an application then you can't afford to rely on logs. You need to be proactive about application performance, finding issues before they snowball into outages that cost you money and customer loyalty.



Your Culture Has Changed

Recent developments in IT processes and culture have an effect on how performance is managed, too. Trends like DevOps represent a shift in how developers and operations collaborate to manage applications, and agile development methodologies have increased the rate of change in many applications. These shift creates a need for tools that 1) make it easy for anyone in the organization to identify performance issues, whether they're an application expert or not, and 2) automatically reflect changes in the application environment. Logging no longer works in this new scenario, because it requires an application expert to know where to look and what to look for. Application performance management tools are necessary to truly enable DevOps at an organization.

Conclusion

Logging has never been a great way to resolve performance problems in a production environment, but recent changes in technology, business and culture have made it even more unsuitable for production monitoring. In order to manage application performance effectively you need a tool that makes it easy for anyone to find and resolve performance bottlenecks no matter where they occur in the application – in other words, you need an approach to managing performance that's designed for the modern world of applications. Logging is simply no longer enough.