An introduction to Application Performance Management (APM)

Application Performance Management, or APM, is the monitoring and management of the availability and performance of software applications. Different people can interpret this definition differently so this article attempts to qualify what APM is, what it includes, and why it is important to your business. If you are going to take control of the performance of your applications, then it is important that you understand what you want to measure and how you want to interpret it in the context of your business.

What is Application Performance Management (APM)?

As applications have evolved from stand-alone applications to client-server applications to distributed applications and ultimately to cloud-based elastic applications, application performance management has evolved to follow suit. When we refer to APM we refer to managing the performance of applications such that we can determine when they are behaving normally and when they are behaving abnormally. Furthermore, when someone goes wrong and an application is behaving abnormally, we need to identify the root cause of the problem quickly so that we can remedy it.

We might observe things like:

- The physical hardware upon which the application is running
- The virtual machines in which the application is running
- The JVM that is hosting the application environment
- The container (application server or web container) in which the application is running
- The behavior of the application itself
- Supporting infrastructure, such as network communications, databases, caches, external web services, and legacy systems

Once we have captured performance metrics from all of these sources, we need to interpret and correlate them with respect to the impact on your business transactions. This is where the magic of APM really kicks in. APM vendors employ experts in different technologies so that they can understand, at a deep level, what performance metrics mean in each individual system and then aggregate those metrics into a holistic view of your application.

The next step is to analyze this holistic view your application performance against what constitutes normalcy. For example, if key business transactions typically respond in less than 4 seconds on Friday mornings at 9am but they are responding in 8 seconds on this particular Friday morning at 9am then the question is why? An APM solution needs to identify the paths through your application for those business transactions, including external dependencies and environmental infrastructure, to determine where they are deviating from normal. It then needs to bundle all of that information together into a digestible format and alert you to the problem. You can then view that information, identify the root cause of the performance anomaly, and respond accordingly.

Finally, depending on your application and deployment environment, there may be things that you can tell the APM solution to do to automatically remediate the problem. For example, if your application is running in a cloud-based environment and your application has been architected in an elastic manner, you can configure rules to add additional servers to your infrastructure under certain conditions.

Thus we can refine our definition of APM to include the following activities:

- The collection of performance metrics across an entire application environment
- The interpretation of those metrics in the light of your business applications
- The analysis of those metrics against what constitutes normalcy
- The capture of relevant contextual information when abnormalities are detected
- Alerts informing you about abnormal behavior
- Rules that define how to react and adapt your application environment to remediate performance problems

Why is APM important?

It probably seems obvious to you that APM is important, but you will likely need to answer the question of APM importance to someone like your boss or the company CFO that wants to know why she must pay for it. In order to qualify the importance of APM, let's consider the alternatives to adopting an APM solution and assess the impact in terms of resolution effort and elapsed down time.

First let's consider how we detect problems. An APM solution alerts you to the abnormal application behavior, but if you don't have an APM solution then you have a few options:

- Build synthetic transactions
- Manual instrumentation
- Wait for your users to call customer support!?

A synthetic transaction is a transaction that you execute against your application and with which you measure performance. Depending on the complexity of your application, it is not difficult to build a small program that calls a service and validates the response. But what do you do with that program? If it runs on your machine then what happens when you're out of the office? Furthermore, if you do detect a functional or performance issue, what do you do with that information? Do you connect to an email server and send alerts? How do you know if this is a real problem or a normal slowdown for your application at this hour and day of the week? Finally, detecting the problem is one thing, how do you find the root cause of the problem?

The next option is manually instrumenting your application, which means that you add performance monitoring code directly to your application and record it somewhere like a database or a file system. Some challenges in manual instrumentation include: what parts of my code do I instrument, how do I analyze it, how do I determine normalcy, how do I propagate those problems up to someone to analyze, what contextual information is important, and so forth. Plus you have introduced a new problem: you have introduced performance monitoring code into your application that you need to maintain. Furthermore, can you dynamically turn it on and off so that your performance monitoring code does not negatively affect the performance of your application? If you learn more about your application and identify additional metrics you want to capture, do you need to rebuild your application and redeploy it to production? What if your performance monitoring code has bugs?

There are other technical options, but what I find most often is that companies are alerted to performance problems when their custom service organization receives complaints from users. I don't think I need to go into details about why this is a bad idea!

Next let's consider how we identify the root cause of a performance problem without an APM solution. Most often I have seen companies do one of two things:

- Review runtime logs
- Attempt to reproduce the problem in a development / test environment

Log files are great sources of information and many times they can identify functional defects in your application (by capturing exception stack traces), but when experiencing performance issues that do not raise exceptions, they typically only introduce additional confusion. You may have heard of, or been directly involved in, a production war room. These war rooms are characterized by finger pointing and attempts to indemnify one's own components so that the pressure to resolve the issue falls on someone else. The bottom line is that these meetings are not fun and not productive.

Alternatively, and usually in parallel, the development team is tasked with reproducing the problem in a test environment. The challenge here is that you usually do not have enough context for these attempts to be fruitful. Furthermore, if you are able to reproduce the problem in a test environment, that is only the first step, now you need to identify the root cause of the problem and resolve it!

So to summarize, APM is important to you so that you can understand the behavior of your application, detect problems before your users are impacted, and rapidly resolve those issues. In business terms, an APM solution is important because it reduces your Mean Time To Resolution (MTTR), which means that performance issues are resolved quicker and more efficiently so that the impact to your business bottom line is reduced.

Evolution of APM

The APM market has evolved substantially over the years, mostly in an attempt to adapt to changing application technologies and deployments. When we had very simple applications that directly accessed a database then APM was not much more than a performance analyzer for a database. But as applications moved to the web and we saw the first wave of application servers then APM solutions really came into their own. At the time we were very concerned with the performance and behavior of individual moving parts, such as:

- Physical servers and the operating system hosting our applications
- JVM
- Application server behavior
- Application response time

We captured metrics from all of these sources and stitched them together into a holistic story. We were deeply interested in garbage collection behavior, thread and connection pools, operating system reads and writes, and so forth. Not to mention, we raised fatal alerts whenever a server went down. Advanced implementations even introduced the ability to trace a request from the web server that received it across tiers to any backend system, such as a database. These were powerful solutions, but then something happened to rock our world: the cloud.

The cloud changed our view of the world because no longer did we take a system- level view of the behavior of our applications, but rather we took an application- centric view of the behavior of our applications. The infrastructure upon which an application runs is still important, but what is more important is whether or not an application is able to execute its business transactions in a normal fashion. If a server goes down, we do not need to worry as long as the application business transactions are still satisfied. As a matter of fact, cloud-based applications are elastic, which means that we should expect the deployment environment to expand and contract on a regular basis. For example, if you know that your business experiences significant load on Fridays from 5pm-10pm then you might want to start up additional virtual servers to support that additional load at 4pm and shut them down at 11pm. The former APM monitoring model of raising alerts when servers go down would drive you nuts.

Furthermore, by expanding and contracting your environment, you may find that single server instances only live for a matter of a few hours. I have heard of one large cloud-based application that uses a very large amount of RAM in its JVMs, but its recycling strategy ensures that those servers are shut down before garbage collection ever has a chance to run. This might be an extreme example, but it illustrates that what was once one of the most impactful performance issues has been rendered a non-issue by a creative deployment model.

You may still find some APM solutions from the old world, but the modern APM vendors have seen these changes in the industry and have designed APM solutions to focus on your application behavior and have placed a far greater importance on the performance and availability of business transactions than on the underlying systems that support them.

Buy versus Build

This article has covered a lot of ground and now you're faced with a choice: do you evaluate APM solutions and choose the one that best fits your needs or do you try to roll your own. I really think this comes down to the same questions that you need to ask yourself in any buy versus build decision: what is your core business and is it financially worth building your own solution?

If your core business is selling widgets then it probably does not make a lot of sense to build your own performance management system. If, on the other hand, your core business is building technology infrastructure and middleware for your clients then it might make sense (but see the answer to question two below). You also have to ask yourself where your expertise lies. If you are a rock star at building an eCommerce site but have not invested the years that APM vendors have in analyzing the underlying technologies to understand how to interpret performance metrics then you run the risk of leaving your domain of expertise and missing something vital.

The next question is: is it financially worth building your own solution? This depends on how complex your applications are and how downtime or performance problems affect your business. If your applications leverage a lot of different technologies (e.g. Java, .NET, PHP, web services, databases, NoSQL data stores) then it is going to be a large undertaking to develop performance management code for all of these environments. But if you have a simple servlet that calls a database then it might not be insurmountable.

Finally, ask yourself about the impact of downtime or performance issues on your business. If your company makes its livelihood by selling its products online then downtime can be disastrous. And in a modern competitive online sales world, performance issues can impact you more than you might expect. Consider how the average person completes a purchase: she typically researches the item online to choose the one she wants. She'll have a set of trusted vendors (and hopefully you're in that honored set) and she'll choose the one with the lowest price. If the site is slow then she'll just move on to the next vendor in her list, which means you just lost the sale. Additionally, customers place a lot of value on their impression of your web presence. This is a hard metric to quantify, but if your web site is slow then it may damage customer impressions of your company and hence lead to a loss in confidence and sales.

All of this is to say that if you have a complex environment and performance issues or downtime are costly to your business then you are far better off buying an APM solution that allows you to focus on your core business and not on building the infrastructure to support your core business.

Conclusion

Application Performance Management involves measuring the performance of your applications, capturing performance metrics from the individual systems that support your applications, and then correlating them into a holistic view. The APM solution observes your application to determine normalcy and, when it detects abnormal behavior, it captures contextual information about the abnormal behavior and notifies you of the problem. Advanced implementations even allow you to react to abnormal behavior by changing your deployment, such as by adding new virtual servers to your application tier that is under stress. An APM solution is important to your business because it can help you reduce your mean time to resolution (MTTR) and lessen the impact of performance issues on your bottom line. If you have a complex application and performance or downtime issues can negatively affect your business then it is in your best interested to evaluate APM solutions and choose the best one for your applications.

This article reviewed APM and helped outline when you should adopt an APM solution. In the next article, we'll review the challenges in implementing an APM strategy and dive much deeper into the features of APM solutions so that you can better understand what it means to capture, analyze, and react to performance problems as they arise.