Recommendations for Evaluating Application Performance Monitoring Tools

Bob Balaban, December, 2015

Introduction and Overview

Application Performance Monitoring ("APM") tools are a great way to gain end-to-end transparency into how well your large-scale applications are performing. They are especially applicable to 3-tier/N-tier architectures supporting large web sites, hosted applications, and "cloud"/SaaS (software as a service) apps.

While the value of a good APM tool is large (when applied appropriately), they can be tricky to install and deploy across many servers and tiers in a large data center. It is therefore important to plan and conduct a significant (lasting at least 2 months) "Proof of Concept" (POC) trial, to fully understand the capabilities, requirements and value to your particular situation.

This article provides guidelines and recommendations for planning and conducting an APM POC, including some specific tips on how to deploy and evaluate . This is not meant to be a deep dive into APM feature sets, but rather a guide to how to evaluate APMs, and how to decide which one is best for your use cases.

Summary Recommendations

Use this basic outline to plan and execute a "Proof of Concept" evaluation of a small number of short-listed vendor tools.

1) Step 1: Pre-POC planning
    a. **Identify stakeholders and beneficiaries** of APM functionality (development, operations, management…).
    b. Determine whether your organization's application and infrastructure is likely to benefit from using an APM tool. Engage stakeholders to **identify likely use cases and desired benefits.**
    c. **Winnow** a list of vendors to a shortlist of 2 or 3. The shortlist should only include vendors whose technology is **directly compatible** with your stack (Java, .NET, etc.), otherwise you won't get deep instrumentation.
    d. Plan to conduct a significant (at least 2 months) **Proof of Concept deployment and evaluation** of the shortlisted vendors.

e. Translate your stakeholders' desired benefits into a **detailed evaluation scorecard** before starting the POC, and revise it as you go. Make sure your scorecard specifically addresses the functionality you most care about. Make sure all your common use cases are covered.

2) Step 2: **Deploy** a POC on a small set of servers (for each short-listed vendor)
    a. Pay attention to deployment details: how hard is it to accomplish? Be sure to **enable all of the available options.** Some tools have features that are only usable after extra setup steps.
    b. **Get training** on client (or portal) use and data analysis from the vendors, troubleshoot real problems and bottlenecks. Get as many people using it as possible. Perceptions of value may vary widely with job function.
    c. Enable as many of your staff as possible to use the tools. Collect feedback from everyone.
    d. Prepare to allocate significant application engineering and ops/devops engineering time to the POC deployment and evaluation.
    e. **Apply the tool(s) for each of your major use cases.** Use it to diagnose real problems and monitor production systems. Evaluate how well the tool addresses each use case, and keep track of high-value features as well as feature gaps.

3) Step 3: Choose a vendor
    a. **Price** should not be a serious point of differentiation until it's time to actually negotiate a purchase.
    b. **Use your scorecard to rate your vendors' products.** Note imbalances in feature sets, evaluate which differences are of major importance to you.

Background: What is APM? Do I Need it?

Application Performance Monitoring

An APM tool is installed in your server/database tiers, and it instruments and monitors the platform (the VM, the operating system) and the application. APM tools collect and report data to a central location, which might be on-premises, or might be hosted by the vendor offsite), where statistics and correlations are analyzed.

Customers typically use these results to monitor the health of their site, to gain insight into who is visiting, and to troubleshoot performance "hot spots" in the application or data tier. Different vendor offerings emphasize slightly different capabilities, though there is a large overlap in functionality across many products.

The most **common use cases** for an APM tool are:

1) Troubleshooting specific performance problems end-to-end, across all tiers.
   Scenario: You get a user report of "slow response time". The APM tool lets you track the user's

transaction path from the browser to the application server to the database. Is the bottleneck in the network? An overloaded apps server CPU? A SQL deadlock?

2) Application health monitoring (green/yellow/red, by server or data center)
   Scenario: You have a large data center with dozens or hundreds of application and database server nodes. You want to have a bird's-eye view of request flow and responsiveness. A good APM tool can pinpoint specific hotspots and monitor various performance metrics. They can raise alarms when metrics violate customized thresholds, or application exceptions are raised. Some APM tools allow for "synthetic testing", where you define a URL path through your application (a "test"), and program the APM system to run that test remotely from several different locations around the world against your site, at scheduled intervals, and report the resulting performance metrics. So, for example, you can find out that your site is performing well from everywhere except (say) China, and then take specific action to investigate and remediate.
3) Database monitoring and hotspot detection
   Scenario: You have application server logging, MBeans and other instrumentation, but you have no visibility into what's happening once a user transaction reaches your database. Which queries or stored procedures are called most often? Which consume the most CPU or take the longest to execute?
4) Dev/Test performance tuning
   Scenario: You want to do performance testing and load testing before you deploy your next application release. Are the configuration or code changes you are about to deploy actually effective? How are your performance benchmarks doing over time? Many APM tools let you create "probes" or "sensors" that can monitor specific pieces of code. Most will also notice and record application exceptions as they happen. You can track individual entry points, and find out how often each is called, and how long each takes to execute.
5) Monitoring application usage and browser metrics
   Scenario: You'd like to collect data on how many individuals visit your site, which pages they browse, and how long they spend there. You also want to know where those visitors are located geographically, what their average response time is, and whether that varies by point of origin. Perhaps you also want to know what browsers, and what versions of each, your visitors are using, so that you can decommission application support for the older and less secure ones.

Target Environments for APM

APM tools are not designed for use with desktop applications. They are architected around "agent" modules that you install on application servers and/or database servers. They work at a low level, instrumenting runtime environments (Java virtual machines, .NET or Python runtimes, and RDBMS executables.

Some APM tools are designed for application server runtimes (e.g., Tomcat, JBoss, .NET), others are specifically for instrumenting database servers (SQLServer, Oracle, DB2, MySQL, etc.). Application agents will report on application heap usage, transaction timings and garbage collection performance. DBMS

agents will report on database CPU and memory consumption, wait times, throughput, disk access, index usage and even query execution plan performance.

Some APM vendors also supply agents specifically for monitoring the operation of the OS on a server node (Linux and Windows are the most common), such as CPU and memory consumption, disk and network i/o. Some vendors collect these metrics using their application server agent.

Many IT organizations will deploy more than one APM tool. You might, for example, choose one vendor whose product is especially strong in the app server tier, but perhaps not especially strong on the database side, and supplement it with another vendor's product built specifically to be strong on database performance. A combination of best of breed tools can give you overall more detail and better insight into what's going on than a one size fits all tool that tries to cover all the bases. The tradeoff, though, is that you lose some ability to track individual transactions end to end, since correlating data from different tools is usually difficult.

APM Configurations and Licensing

There are generally two different deployment and licensing models for APM tools: on-premises and "cloud", or software as a service (SaaS). It is important to understand the pros and cons of each (not all vendors offer both options).

On premises: The software is licensed and deployed in the customer's data center. The APM agents, the data repository and the user access portal all need to be configured and run on customer hardware. On the plus side, all performance and other data generated by the tool remain in the customer's hands.

SaaS/cloud: This is usually licensed as a subscription, where the customer pays a fee per month to use the software. The APM agents are installed in the customer's data center, but report their collected data back to the vendor's central repository. Some vendors offer a hosted model, where each customer's data goes to a dedicated server, while others implement a multi-tenanted shared model. In either case, the vendor will usually host a web site or portal for the customer to access, and will maintain the customer's data.

There are two important configuration considerations to weigh before choosing an APM vendor:

- On-premises deployment means your data don't leave your control, but you have to manage the APM tool locally. If an on-premises tool is deployed directly in a production environment, it may be the case that not everyone will have access to the data, for example, if development personnel are not allowed direct access to production servers.
- Cloud deployment means you don't have to administer the APM software or access portal, but you need to be comfortable with potentially sensitive data travelling outside your firewall to a vendor's system. Detailed database queries, for example, might contain sensitive user information embedded in the SQL. If this is a concern, get your potential vendors to commit to a security plan that omits or masks such data before going ahead with a POC.

Tips On Deploying APM Agents

"Ease of deployment" should definitely be a metric that you evaluate on your vendor scorecard. The amount of effort to deploy an APM tool widely (perhaps across hundreds of server nodes) can vary by an order of magnitude from one vendor's product to another. You might not weight this particular item heavily – after all, once it's deployed, it's deployed, although someone will still have to apply (and sometime re-apply updates and other fixes or configuration changes.

Here are some best practices for installing APM agents on your servers:

1) **Automate** as much of the process as possible. Some vendors supply you with a runnable install kit (a .msi for Windows, for example), others supply only a .zip containing all the required files. In both cases, however, you will need to do some amount of customization for each server node. At a minimum, you might have to set up a configuration parameter with a unique name for the server. Or, you might need to supply an IP address (within your data center) for a "collector" service that the agent(s) will report to.

   You will want to pre-configure as much of the setup as possible before you copy the kit to the server. Generally, this means creating a separate install "package" for each node (or maybe only individual XML config files that you can overlay on the install kit's files after expansion).

   You will also want to replace any required manual editing that is needed on the server with automation scripts. With Tomcat application servers, for example, most APM Java agents are loaded into the Tomcat JVM process via a command line option in the Tomcat startup. You want that type of configuration change to be handled by a batch script, not by manual editing at install time, where a fat-fingered mistake can cause real trouble.

2) **Practice** your installation setup on a non-critical server that you fully control, until you are completely happy with the process. Then turn it over to an Ops engineer for a practice run on a production server, with you looking over his or her shoulder.

3) **Document** the installation process completely and in detail, including all the steps you might think are obvious. For example, don't assume a junior ops engineer will know how to set the system PATH (without breaking it), or how to unzip a package to the correct location. These steps will (ultimately, once you buy one of the tools) have to be done over and over, at least once on each server node in your cloud.

   Also plan to document any surprises (positive or negative) that you encounter, you'll want to add these to your scorecard.

4) **Enable every possible feature.** With some tools, some features are only available if you take extra steps at agent install time. Because you hope to have a diverse selection of tool evaluators, you may not know in advance which features will be important to which users, so

just turn them all on. Just be aware that this may impact your install process (e.g., adding JARs to the CLASSPATH, extra JVM command line options, etc.).

5) **Install on multiple servers.** While you can glean useful data from running an APM tool on a single application server or database server, the really useful experience in setup and troubleshooting can only come from a multi-node deployment, preferably in a production environment.

   You should plan on deploying the tool in both pre-production and production environments, since your use cases will be different for each of these.

Tips On Conducting the Evaluation

There's a lot to be learned just from doing the agent installations, but of course the whole point of the project is to see which vendor will address your real-life problems most effectively.

Here are some best practices for planning and conducting a useful APM evaluation:

1) **Evaluate at least two vendors.** A side by side comparison will be much more useful to you in clarifying which use cases are most important, and which tool features work best for you. If you have the capacity, evaluate 3 (more than 3 probably won't provide incremental intelligence, and will be a lot more work).

2) **Get training.** These tools are very powerful, and it's not always obvious how to find the feature or features that you're looking for. Ask the vendor to provide training, either onsite or online. Ask for multiple training sessions over a period of a few weeks. Once you start collecting real data, you will accumulate lots of specific "how do I?" and "what does this mean?" type questions; the vendor should be more than happy to give you specific assistance.

3) **Get as many evaluators engaged as possible.** Your set of use cases is probably pretty diverse, and your community of tool evaluators should be diverse as well. People with different job functions (development, QA, DevOps, Ops, Professional Services) will weight features differently (and may even express widely varying use cases); you want to collect ALL possible feedback and make the selection that serves the maximum number of use cases.

4) **Update the scorecard as you go.** You will find as you do real work with the APM tool(s) that some features (even use cases) you hadn't thought of before are actually quite important to you. You will also find the opposite: some features that you thought would be critical when you started turn out to be uninteresting later. Keep the scorecard updated, and share it with your evaluator community (and even with your vendor candidates, why not?)

5) **Log all tech support calls to the vendor.** The number and content of the calls that get made to the vendor to report problems and ask questions tells you two things: how engaged your

evaluator community is, and how responsive and helpful the vendor is. Both are valuable inputs to your decision making process.

6) **Make the decision based on functionality first, price second.** Price is always negotiable, product functionality usually is not. It's likely that a side by side vendor comparison will result in a clear choice of one tool over the other based purely on your scorecard's evaluation criteria (primarily, functionality, ease of use, etc.). Once you've made a first-round pick based on overall usefulness, then you can start negotiating the price. Even in cases where one vendor's product is much costlier than another's, there is almost always room to negotiate the price down. It can also be useful at this point in the process to let both your number one and number two choices know that they are not the only vendor in the game.

   Once you've made your purchase decision, be sure to instruct the vendors who didn't make it to the finish line to delete all your APM data. They have no reason to keep it once the evaluation is over.

Conclusion

APM tools are complex beasts, and vary widely one from another, but the payoff from using them effectively is very high, especially for large, distributed applications and web sites.

The odds of picking the right one for you go up significantly if you invest in planning and executing an in-depth Proof of Concept (or Proof of Value) test deployment. Planning carefully, engaging the right stakeholders and executing well will go a long way to ensuring a good purchasing decision that will yield strong returns over several years.