

5 Critical Strategies for Application Performance Management

Business White Paper

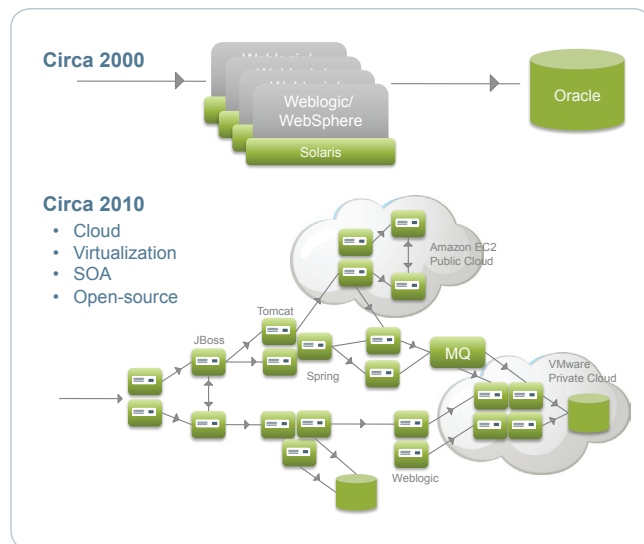
October, 2010

Introduction

Responsibility for overseeing and managing applications is increasingly moving away from application architects and developers. Rather, it is IT operations and infrastructure professionals who have begun to shoulder the responsibility for ensuring application health and performance.

This is particularly true in regards to distributed applications. A fundamental change in the way that applications are built, deployed, and scaled has caused web applications to shift from being monolithic in nature to many components distributed over multiple tiers and nodes, all connected through service-oriented architectures. This trend has contributed to IT operations and infrastructure professionals being the ones whose “beeper goes off” when applications fail to meet business SLAs.

Therefore, it’s more important than ever that tools designed for Application Performance Management (APM) enable IT operations and infrastructure professionals to locate and resolve problems without necessarily calling in the development team for assistance. APM tools must be extremely intuitive, far-ranging in their capability, and able to speak the language of business—rather than the language of developers.



In particular, APM tools must be selected with the five following strategic end goals in mind:

1. Simple to install and use
2. Able to go wide
3. Able to go deep
4. Able to understand business transactions
5. Built for cloud environments

Without all of these capabilities, an APM tool may be able to support certain application performance functions but will ultimately be ill-equipped for the demands of a quickly changing IT environment.

1. SIMPLE TO INSTALL AND USE

Many APM tools are difficult to install, use, and maintain. But within the APM space, there are standard practices available, such as the ability to auto-discover an application’s architecture and quickly alter the mapping of the application’s topology when agile release cycles introduce code changes. Your APM tool should leverage those best practices and make application management extremely easy for you.

For example, an APM tool should not require more than a series of simple steps in order to be installed. It should be up, running, and instrumenting the distributed application within hours or even minutes. The process looks something like this: the end user installs the agents on all managed servers and virtual machines, installs the controller, and re-starts the application. At that point, the tool itself should be able to handle the challenge of mapping all the databases, tiers, and nodes within the distributed application, as well as displaying those relationships in an intuitive and visual way.

However, with regards to many APM solutions, it's necessary to perform a great deal of "post-install" work—mostly aimed at tying pieces of the data flow together, manually mapping the relationship between the JVMs, setting up instrumentation for visibility, preparing the aggregation for all the data collectors, and identifying the logical tiers. Unless the APM tool is discovering this information for you, the "simple implementation" quickly turns into a mountain of post-install work.

Questions to ask an APM provider:

- What is the installation process up front?
- What kind of post-install work is required?
- How does the tool map the application?
- How does it display that mapping?
- Does it keep up with the needs of agile development?
- How much manual instrumentation is needed for complete visibility?

2. ABLE TO GO WIDE

Traditional applications often consisted of little more than a few JVMs talking to one another. In the present day, a typical IT environment consists of a multitude of open source and proprietary components and distributed tiers, all attempting to communicate together in order to perform complex business transactions.

If an APM tool captures just a piece of this massive web of communication—but is unable to reveal the entire architecture, leaving blind spots in place—then the solution will fail to provide value. At the same time, the tool should not just provide visibility into what already exists, but be able to automatically discover, trace, correlate, and visualize transaction performance when agile development cycles introduce new code into the environment.

Questions to ask an APM provider:

- Does your tool have insight into the entire distributed environment?
- Are there any blind spots in your current architecture?
- If it does map the environment, how does that data come back to you? Is it simple to view?

3. ABLE TO GO DEEP (AND THAT MEANS IN PRODUCTION TOO)

New tools must be introduced into the production environment with extreme caution, care, and planning. That's why many APM tools aren't used in production, even if they're adept at illuminating a highly distributed application. The ability to go "wide" often precludes the ability to go "deep" in production, where adding too much overhead remains an ever-present danger.

And yet, without the ability to go deep on demand—to isolate a problem at the code level without calling in the developer who wrote the code—the APM tool won't be a reliable asset. It's necessary to drill down and see method-level detail of a particular transaction, and understand whether the incident is caused by resource constraints, bad code, or any number of potential culprits.

How can the tool retain the ability to "go deep" but still be "always on," ensuring that the diagnostic process does not result in significant overhead? Clearly, this means something other than manually instrumenting all the classes and the code paths—which by itself could still result in blind spots, and will very likely result in unacceptable overhead. Rather, the APM tool must be able to capture details of a problem whenever a problem occurs. This sounds glaringly obvious, but many APM tools work poorly in production as a result of dragging down the application with a lack of "always on" diagnostic capabilities.

This is a key point: an APM tool needs to go both deep and wide in production, being "always on" but without bringing significant overhead to bear on the application. If the tool causes more than 2% overhead, it will not be usable by anyone other than the development team.

Questions to ask an APM provider:

- Can your APM tool go deep as well as wide?
- How does it isolate problem areas at the code level?
- How much overhead does it add? Less than 2%?

4. ABLE TO UNDERSTAND BUSINESS TRANSACTIONS

The ability for an APM tool to focus on business transactions is important. It allows the tool to create a common language between developers and IT operations by representing the transaction, rather than a snippet of code.

A business transaction represents a “user generated” action. For example, a user might add a book to the check-out cart, or a hiring manager might pull up an online resume. The APM tool needs to be able to make these actions highly visible to the IT operations team. This is an essential part of the simplicity and usability of the APM tool: the ability to talk in the language of business.

Once the APM tool gives you a clear view into the application’s business transactions, then it’s possible to measure the performance of those transactions. In legacy APM tools, it’s necessary for you to do this by setting manual thresholds for each transaction. For example, you might say that the check-out transaction takes two seconds on average, or the user log-in takes half a second.

The problem with this approach is that you may not have the data to set those thresholds yourself. You could make an educated guess, but the burden is on you to tell the APM tool how well your application performs. If the transaction responds to load differently on Sunday versus Monday, or at 9 a.m. versus 8 p.m., it’s up to you to specify the appropriate threshold for each time period. If your performance policies aren’t granular enough to reflect the true performance profile that occurs during the hours in which you operate, as well as to account for periodic variations, you’ll either lack visibility or be flooded with false alarms.

An APM tool that leverages best practices should be able to set those thresholds for you. This is often called dynamic baselining. This means being able to set baselines for your application by discovering how each transaction’s performance may vary over specified operating periods. It observes periodic variations, accounts for them, and sets thresholds accordingly. A tool that sets dynamic baselines for each business transaction will be highly accurate and eliminate false alarms.

Questions to ask an APM provider:

- How do you define “business transactions”?
- How does your tool represent them?
- How does your tool speak the “business language” of the application?
- Does the tool learn and react from performance baselines dynamically? If not, how are they set and maintained?

5. BUILT FOR CLOUD ENVIRONMENTS

Many businesses are eyeing the cloud as the next important initiative for their critical business applications. They understand the need for business agility—the need to dramatically ramp up capacity without pouring tons of budget into their physical data center. In addition, they plan to release new services that are capacity-intensive, and which require the ability to provision hundreds or even thousands of cloud nodes quickly.

Take the example of Netflix. As one of the most forward-looking companies in regards to new IT initiatives, they have moved many of their mission-critical processes to the cloud. This allows them to gain elastic capacity in the face of unpredictable demand spikes. It also allows them to focus on innovation and differentiation, rather than having to waste time performing mundane tasks in the physical data center.

Ordinary, everyday data center tools simply don’t work in the cloud. The cloud landscape changes everything:

There are more things to manage by a factor of 10. Whereas the physical data center may have had 40-50 megaservers in the past, cloud nodes are made up of thousands of commodity, low-cost servers. Thus, an individual server means less. Managing application performance and availability through measuring the health of servers (CPU utilization, memory utilization) is no longer a reliable proxy for application health.

The environment is dynamic, rather than static. That means there's no longer the same set of megaservers serving traffic each and every day. Cloud servers are easily replaced, and thousands of instances can be added or dropped in a minute. Thus, any concept of management that relies on a static set of servers, connections, or agents is severely outdated; the lifespan of a node may be 5 days or less!

How can an APM solution be built for this new kind of environment?

Scale scale scale. The APM solution must be able to monitor thousands of cloud nodes from a single management server in order to provide end-to-end transaction performance metrics and tracing. If the APM solution can only scale to 200:1, then an IT operations professional will require multiple consoles and will lack "a single pane of glass" to monitor application performance.

Ability to be dynamic. The APM solution must be able to handle hundreds of nodes being provisioned and de-provisioned, which means the APM tool's performance monitoring, metrics, transaction tracing, service dependency modeling, and deep diagnostics all need to work in an extremely dynamic environment. Legacy APM solutions that don't dynamically adapt to infrastructure changes will become useless quickly.

Capacity provisioning. Finally, a cloud-ready APM solution won't simply be able to map and monitor a cloud-based application, but also assist IT professionals in dynamically provisioning resources. By scaling up or scaling down capacity as needed, the tool is not simply identifying application problems but resolving them in an "always on" state.

Questions to ask an APM provider:

- Is your tool built for a cloud environment "ground up," or has it been retrofitted?
- Can the tool manage application performance with respect to virtual and clouds environment? If so, how?
- How many nodes can the tool monitor in the cloud?
- How does it adapt to the dynamic nature of the cloud?
- Can it dynamically provision resources across the cloud?
- What cloud-ready environments has it monitored in the past? What are the customer names?

SUMMARY

Managing the performance of your application requires a strategic approach, particularly in a space crowded with multiple application performance solutions boasting very similar messages. To assist your decision, leverage these 5 strategies for application performance management—and ensure the ongoing health and reliability of your revenue-critical applications.

ABOUT APPDYNAMICS

Founded in 2008, AppDynamics is a next-generation Application Performance Management (APM) company that delivers rapid problem resolution for highly distributed applications through easy-to-use transaction flow monitoring and deep diagnostics. Unlike other APM providers, AppDynamics finds the root cause of performance problems without introducing excess overhead or requiring a complex and costly installation. AppDynamics is also the first APM provider to dynamically scale applications in virtual, physical, and cloud environments. For more information, visit www.appdynamics.com or download our free java performance tool at www.appdynamics.com/free.



AppDynamics
303 Second Street, Suite 450 | San Francisco, CA 94107
www.appdynamics.com