EXECUTIVE BRIEF | APRIL 2016

Next-Generation Performance Testing with Service Virtualization and Application Performance Management

Summary

For organizations that increasingly rely on technology to provide value to stakeholders improving application delivery capability is critical to remaining competitive and relevant.

Current approaches for predicting with a high degree of confidence the performance of an application prior to implementation into production can be antiquated, expensive and time consuming. This document outlines a next generation performance management approach being adopted by progressive organizations around the world.

Adopting a next generation performance management approach can offer a range of benefits including efficiency gains in application delivery, cost savings, improved agility and better performing applications.

Background

Performance has always been an important component of application quality, but the growth in online channels of interaction means that for many organizations, performance is now critical. Poor performance can impact customer experience, revenues, profitability, brand, and in some cases, the viability of organizations.

The importance of performance has increased at the same time as application workloads have grown larger and more unpredictable; unfortunately, there have been few corresponding improvements in the tools and techniques for predicting and tuning application performance. This paper examines some of the issues that may be associated with traditional performance management techniques and describes how solutions from CA Technologies can help address these issues.

Performance Management: Current State

Application performance is typically managed through a combination of end-to-end performance testing and application monitoring. The intent of application monitoring is to detect performance problems that occur in the production environment and provide information to assist support teams to diagnose and resolve the root cause of issues before end users are impacted. Application monitoring capabilities vary greatly with some organizations having almost no capabilities while others have very mature approaches. What tends to be consistent across all organizations is that there is very little integration between application monitoring and end-to-end performance test functions.

End-to-end performance testing is currently one of the approaches organizations use to proactively ensure application performance. This activity typically occurs at the end of the delivery lifecycle in an environment that attempts to replicate production and is usually performed by using a load testing tool to simulate a large number of concurrent users accessing the application's user interface. While common, this approach can have a number of issues: it can only be done late in the lifecycle, requires large amounts of infrastructure, seldom replicates production to the appropriate level, fails to allow the majority of negative testing conditions, and is challenging to use to provision appropriate test data sets.

Lifecycle Stage

Traditional performance testing typically requires that all components of the application are "completed," integrated and deployed into an appropriate environment. These requirements mean that this testing usually occurs near the very end of the delivery lifecycle. Late execution of performance testing generates a number of challenges, including:

- **Shrinking windows.** Software delivery projects have a tendency to slip schedules; as a result there is often pressure to reduce the time spent on late lifecycle activities such as performance testing. This effect is compounded by the unpredictable nature of project delays. As performance testing is often carried out by a centralized center of excellence, a delay in one project can impact the windows available for multiple projects.
- **Cost of quality.** The longer a defect goes undetected, the more it can cost to repair—this is particularly true of performance defects, which are often a consequence of the application architecture. Performance testing late in the lifecycle means that performance issues are not detected until the last minute, fixes are typically very expensive and may require staff that have long since moved to other projects.
- **Schedule conflicts.** Testing of multiple channel applications that access shared EIB/middleware and system of records can cause scheduling constraints unless you have a mirrored configuration of your production environment available for performance testing including separate networks and network components, as well as the rest of the application infrastructure up to and including your mainframes. This approach can be very expensive to build and support, and still will not allow for the simulation of System of Record, partner or database slowdown conditions.

- **Overlapping releases.** Overlapping SDLC releases add complexity to the ecosystem in that you need the capability to test multiple code releases concurrently. The ability to fund and support multiple, end-to-end performance testing environments is just not prudent for most large organizations.

Infrastructure Requirements

End-to-end performance testing needs to occur in an environment that replicates production. The costs associated with providing these environments have grown rapidly over recent years and look set to escalate even further. These cost increases are driven by a number of factors, including:

- Application workloads continue to grow rapidly, requiring increased amounts of infrastructure to support those workloads.
- The evolution of composite application architectures means that applications are often made up of a large number of component applications. Even a relatively simple service might require a significant subset of an organization's application portfolio to be assembled to support an end-to-end test.
- The increased adoption of application programming interfaces (APIs) and microservices means that many applications are dependent on services provided by third parties, which may charge additional fees for access to test interfaces. Moreover, the availability and performance of those interfaces may not be identical to the production versions.
- The high costs involved in the provision of infrastructure for end-to-end performance testing mean that most organizations have very few suitable environments and a high degree of contention for those environments. Cost and complexity also mean it is increasingly common for end-to-end test environments to differ substantially from production, further increasing the risk of production performance issues.

Test Data Management

The management of performance test data for modern applications presents a number of challenges. Composite applications typically have multiple data stores that need to be provisioned with test data and synchronized. Higher workloads mean that the volume of data required for a realistic test is greater than ever before. In addition, privacy regulations mean that the use of production data for performance testing is no longer an option for many organizations.

Test data management is typically the most costly and time-consuming activity involved in running a performance test.

To complicate test data management further, a fair number of tests are destructive—thus the need to reset, or condition, the data for the next test adds time and complexity to the schedule.

Next-Generation Performance Management

As outlined above, many existing approaches to ensuring applications deliver adequate levels of performance are flawed. Performance problems are typically not detected until very late in the delivery lifecycle, at best, or in production, at worst. In addition, the costs and time taken to detect and resolve performance issues are often unacceptably high and continue to grow.

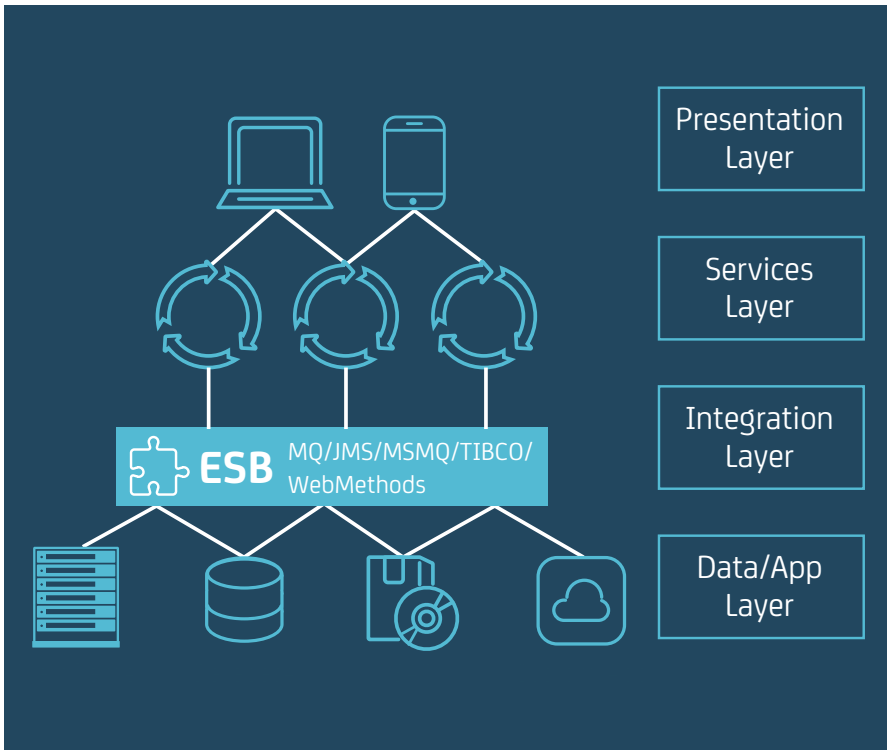
Fortunately, there is an alternative approach for organizations wishing to address the growing cost and ineffectiveness of traditional approaches to performance management. A number of organizations are finding that component-based performance testing coupled with advanced Application Performance Management solutions from CA Technologies can address these issues.

Component-Based Performance Testing

Modern applications are typically component-based: application components leverage services provided by other components often via an ESB or integration layer as illustrated in the diagram below. Note that this diagram is very much a stylized representation of a component-based architecture and typically interactions between the various components are much more complex than indicated below.

Figure A.

Component-based architectures



Component-based architectures are both a challenge and an opportunity. They are a challenge as they contribute significantly to the cost of infrastructure; however, they do present us with the opportunity to test performance at the component level. Component-level performance testing is where we only test the component(s) that have changed; this approach is potentially beneficial for a number of reasons:

- Smaller scale component level tests are much easier and cheaper to run than full end-to-end tests.
- Testing can commence once the component is available, which usually occurs much earlier in the lifecycle than availability of the full system.
- Often, only a few components change. Component-based testing lets teams focus on testing changes, as opposed to end-to-end testing which tends to "dilute" testing effort. For this reason, component-based testing is much more likely to find defects.
- It provides the ability to find the true break point of a given application.
- Component-level testing allows for the isolation of the changing application so that we can focus more deeply in testing of:
  - Resiliency testing for fail over
  - The impact to the application of a service provider slowdown
  - Exercising capacity above our current peak production numbers
  - Operational testing of alerts, monitoring triggers and production automation scripts prior to implementation of the changes into production

While component-based testing is an attractive concept, it has been very difficult to actually implement in practice. There are typically two main barriers: the ability to isolate a component and the intelligence around a component's interactions.

Isolation refers to our ability to run a component or group of related components without requiring access to their dependencies. For example, consider the ESB component in Figure A; this has dependencies on all of the lower tier components and cannot be run or tested without those components being available. We would ideally like to test the ESB without having to make all these dependencies available.

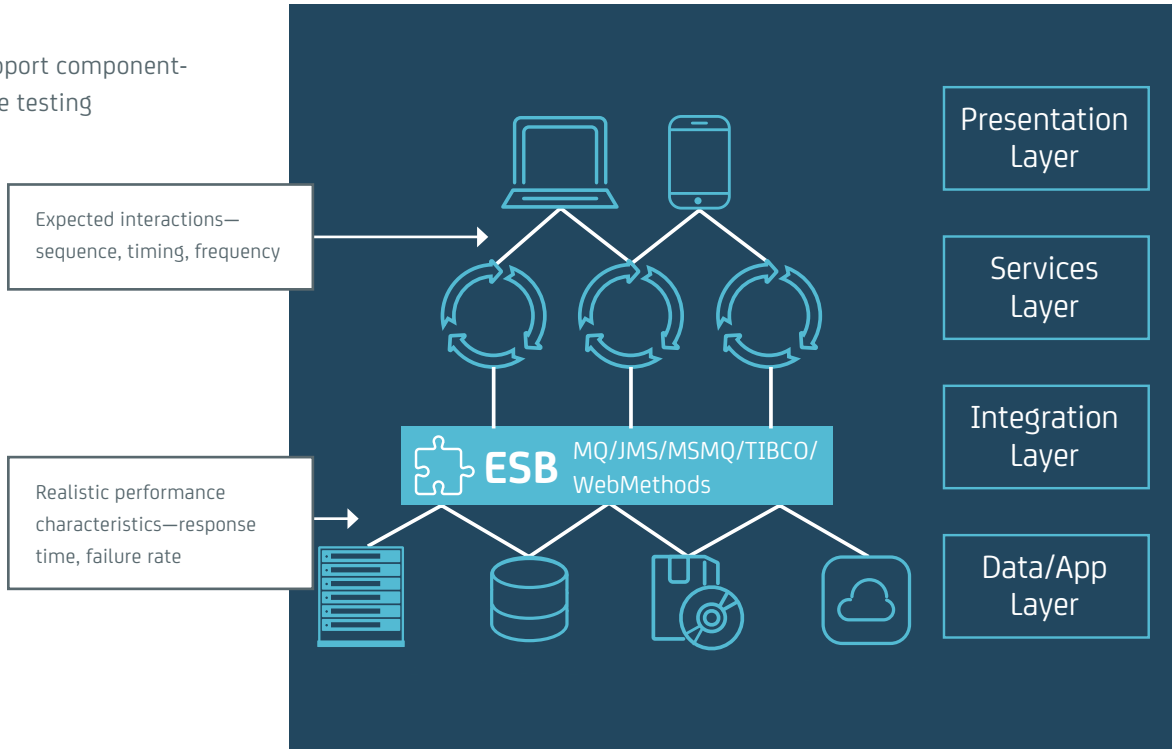
Many organizations make use of hand-coded stubs to isolate components. A stub is a simple piece of code that simulates the behavior of a component. While stubs are often useful for development purposes, they are labor intensive to create and usually not well suited to performance testing. In order to be able to use stubs for performance testing, we need to be able to control how they perform. The developers who typically build stubs are usually not interested in modeling performance, and doing so would greatly increase the cost of stub creation and maintenance.

To implement component-based performance testing, we need a method of rapidly building intelligent stubs that are capable of simulating a range of different performance scenarios.

We also need intelligence about the behavior of the component under test and its dependencies; this intelligence is used to design our performance scenarios and to tune the behavior of the intelligent stubs.

Figure B.

Intelligence to support component-based performance testing



As an example, consider a scenario where we would like to run a performance test against the ESB component shown in the diagram above to determine whether some changes to existing services have impacted performance. We would like to make this test as accurate as possible so we need to understand what ESB services are typically called and the sequence and frequency of these calls. If we are to replace the downstream dependencies with stubs we need intelligence on how those dependencies perform to ensure our testing is realistic.

Component-Based Performance Testing from CA Technologies

CA Technologies can provide organizations with a comprehensive solution designed to address the issues around isolation and intelligence. CA Service Virtualization addresses the problem of providing intelligent stateful stubs through its ability to observe then simulate the behavior of dependent systems. The use of service virtualization can reduce or eliminate a large amount of manual effort and produces stubs which can be used in many lifecycle phases, including performance testing.

CA Application Performance Management (CA APM) is able to monitor production application performance at all tiers and feed the gathered intelligence back to the performance test team to support component-based testing efforts. Integration between the APM and service virtualization offerings greatly assists the performance test team in acquiring and utilizing the gathered intelligence.

The integration between these offerings delivers some other areas of value that might be exploited by a performance testing group. Data gathered from the production systems by CA APM can be used to automatically generate test cases in CA Continuous Application Insight, another module in the DevTest suite from CA Technolgies. Automatic generation of test cases for use in functional or performance testing may well justify supplementing or replacing existing test automation solutions.

In addition, the ability of CA APM to monitor behavior across and within application tiers is very useful to the performance testing and development teams. The root cause of many performance related problems detected in testing can be quickly and easily identified through the use of CA APM.

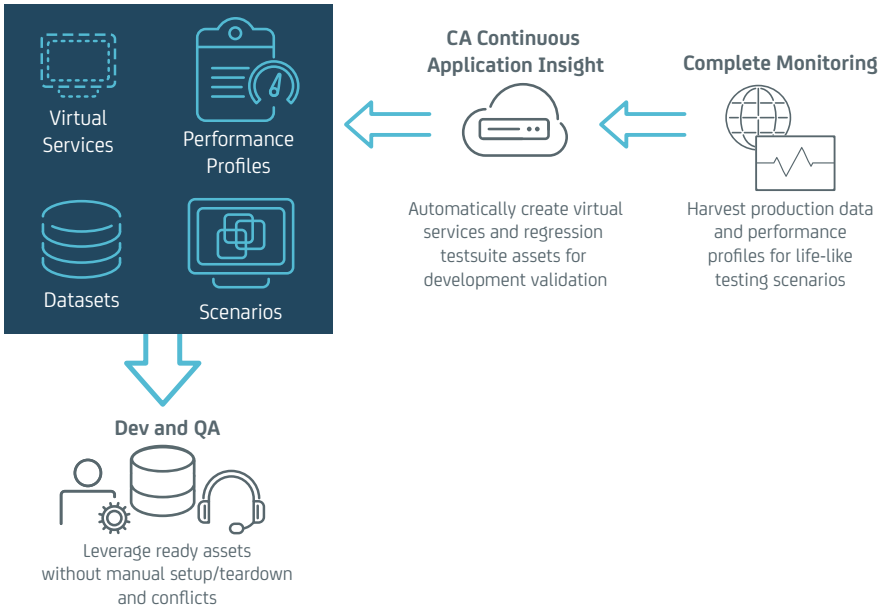
Figure B provides a high-level view of the components involved in the next-generation performance management approach. Production data is mined by CA APM and used to provide intelligence to the service virtualization and test components for CA Continuous Application Insight. This intelligence can then be used to performance test a specific component in isolation, thus minimizing the time and infrastructure required for testing.

Additional benefits

This document has focused on the use of CA Service Virtualization combined with CA APM to enable component-based performance testing. The value of these solutions is far greater than the specific uses outlined in this document. More information on these solutions can be found at ca.com/sv.

Figure C.

Next-generation performance management



Adopting Next-Generation Performance Management

There are a number of organizations around the world that have adopted or are in the process of adopting next-generation performance testing. The most mature organizations never perform end-to-end performance tests and typically leverage service virtualization for functional testing and development purposes as well.

Mature organizations can often see a number of benefits from the adoption of next-generation performance management, including:

- Reduced infrastructure costs through virtualizing expensive dependencies
- Reduction in time taken to set up and manage testing environments as a result of fewer "moving parts"
- Reduced test data management effort (Virtualizing dependencies means that only interface data is required, not full dependent system data stores. Interface-only data is easier to create and synchronize with other systems.)
- Improved quality as a result of being able to focus testing on changed or new components
- Ability to concurrently test multiple applications within the ecosystem
- Ability to concurrently test multiple CODE BASES within the same time frame
- Deeper level of resiliency testing, which is critical to the production operations

Adopting these practices and achieving these outcomes is not something that can be achieved overnight. Supporting technology must be deployed, people trained, assets and procedures created, and trust earned.

In most organizations, there are opportunities to start small and build up the necessary competencies, while delivering a number of quick wins. Service virtualization can be deployed as a point solution to address the unavailability of key systems in preproduction. CA APM can be deployed in parallel to initially address the needs of operations groups. As APM data becomes available, it can be used to drive the increased adoption of service virtualization and component-based testing.

As applications become increasingly more complex and reliant on component-based architectures and third-party cloud systems, the traditional performance testing approach of building a production "replica" will become more and more difficult. For most organizations, the move to next-generation performance management is probably inevitable.

Connect with CA Technologies at ca.com

CA Technologies (NASDAQ: CA) creates software that fuels transformation for companies and enables them to seize the opportunities of the application economy. Software is at the heart of every business, in every industry. From planning to development to management and security, CA is working with companies worldwide to change the way we live, transact and communicate – across mobile, private and public cloud, distributed and mainframe environments. Learn more at **ca.com**.

CS200-193521_0416