Stephan Kehren
(stephan.kehren@neotys.com)
verantwortet bei Neotys als Senior Performance-Ingenieur den Bereich „professional services" für die deutschsprachigen Länder Deutschland, Österreich und Schweiz (DACH). Er arbeitete in leitender Position an vielen großen Testprogrammen in verschiedenen Branchen, von Legacy-Mainframe zu SAP und allen Arten von Web-Anwendungen.

Modern Load Testing
Einführung in moderne Lasttests

Performance-Tests der „alten Schule" erfüllen nicht mehr die Anforderungen der modernen Anwendungsentwicklung. Anwendungen werden zunehmend komplexer und weisen kürzere Entwicklungszyklen auf. Dies erfordert neue, agile Entwicklungs- und Testmethoden. Die Anwendungsleistung als Teil der umfassenden Benutzerfreundlichkeit ist heute einer der Schlüsselaspekte der Anwendungsqualität. Sequenzielle Projekte der „alten Schule" mit statischen Qualitäts-, Implementierungs- und Testphasen, die Performance- und Lasttests bis zum Projektende hinauszögern, können daher die Performance gefährden.

Einer der Hauptfaktoren für diesen Wandel hin zu modernen Lasttests ist auf die wachsende Komplexität der IT-Welt zurückzuführen:

- Die meisten Anwender nutzen mobile Geräte, Thin Clients, Tablets und andere Geräte.
- Es werden komplexe Architekturen errichtet, die von mehreren Anwendungen gleichzeitig genutzt werden.
- Neue Technologien bieten verschiedene Lösungen an (Ajax-Framework, RIA, WebSocket und viele mehr), die eine höhere Benutzerfreundlichkeit der Anwendungen ermöglichen.

Die Testmethodik muss dieser Komplexität und der Entwicklungsmethodik folgen. Zudem sind maßgebliche Faktoren nicht mehr nur die Qualität, sondern auch die Endkundenerfahrung. Letztere ist eine Kombination aus Optik und Handhabung, Stabilität, Sicherheit und Performance. Eine unabhängige Untersuchung* zeigt, dass schon 1 Sekunde Wartezeit 8 Prozent weniger Kunden bedeutet, bei 3 Sekunden werden 40 Prozent der Kunden verloren, viele davon für immer.

* IDG Connect führte, im Auftrag von Neotys, eine Befragung in USA und Europa durch und hat die Ergebnisse ausgewertet.

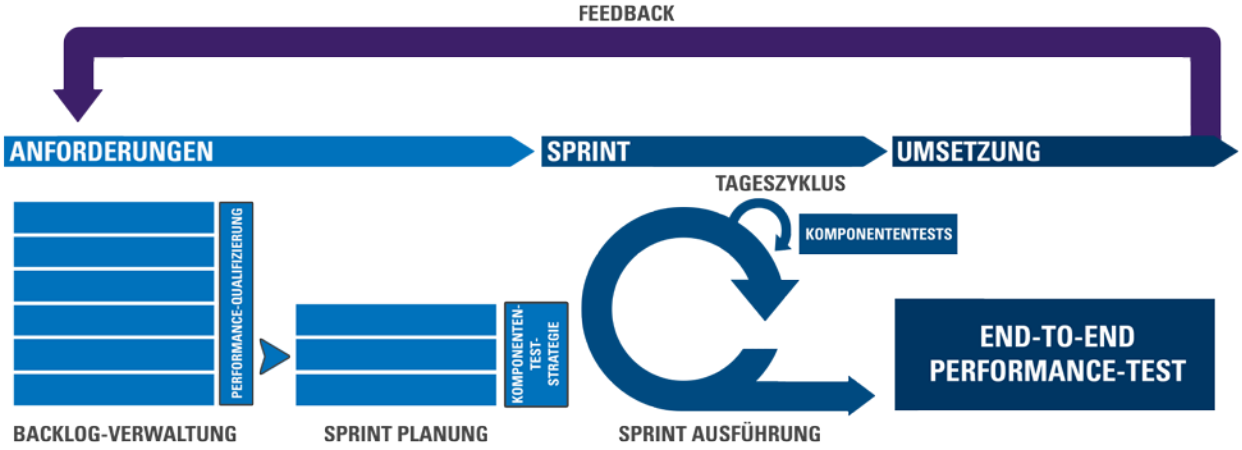

Abb. 1: Ausführung eines Lasttests im Anschluss an einen Sprint

Wenn eine Anwendung Performance-Tests erfolgreich besteht, im Produktivbetrieb jedoch scheitert, so ist dies häufig auf unrealistische Testverfahren zurückzuführen.

Phasen eines Lasttest-Projekts

Mit aktuellen Entwicklungsmethoden wie Agile und DevOps lassen sich Anwendungen erstellen, die rasch auf die Kundenbedürfnisse eingehen. Diese Methoden erfordern eine Modernisierung der Projektorganisation sowie eine enge Zusammenarbeit der beteiligten Teams.

In einem modernen Projektlebenszyklus kann die Prüfung der Performance nur dann in einem frühen Stadium durchgeführt werden, wenn einzelne Komponenten nach jedem Erstellungsprozess getestet werden und unmittelbar nach Assemblierung der Anwendung ein durchgängiger Performance-Test implementiert wird (siehe Abbildung 1).

Festlegung einer Performance-Test-Strategie

Dies ist der erste und wichtigste Schritt eines Performance-Tests. Damit werden folgende Elemente definiert: (1) der Umfang und die Zielsetzung des Performance-Tests, (2) die Lastrichtlinien und (3) die SLA (Service Level Agreements).

Nach Ermittlung der für Performance-Tests erforderlichen Funktionsbereiche müssen Sie die Geschäftsschritte in technische Workflows aufgliedern, die zur Veranschaulichung der technischen Komponenten dienen.

Nach jedem Sprint muss die assemblierte Anwendung anhand von realistischen Benutzertests überprüft werden (durch Einbeziehung mehrerer Komponenten). Auch wenn einzelne Komponenten bereits getestet wurden, müssen unbedingt nachfolgende Parameter gemessen werden:

- das Verhalten des Systems bei paralleler Ausführung mehrerer Geschäftsprozesse,
- die Antwortzeit für den echten Benutzer,
- die Verfügbarkeit der Architektur,
- die Dimensionierung der Architektur und
- die Caching-Richtlinie.

Der Testaufwand wird mit fortschreitender Projektdauer immer komplexer.

Es ist wichtig, dass die Testumgebung die Produktionsumgebung so gut wie möglich widerspiegelt; einzelne Unterschiede sind aber durchaus möglich.

Modellierung von Performance-Tests

Ziel der Lasttests ist die Simulation des realistischen Benutzerverhaltens an der Anwendung. Wenn nicht repräsentative Schritte eines Benutzers ausgewählt werden oder wenn nicht die entsprechenden Lastrichtlinien festgelegt werden, kann das Verhalten der Anwendung unter Last nicht korrekt geprüft werden.

Das Modellieren von Performance-Tests erfordert in erster Linie Zeit, um die Anwendung vollständig zu verstehen. Zu berücksichtigen sind:

- Wie arbeiten die Benutzer am System?
- Welche Gewohnheiten haben sie?
- Wann und wie oft verwenden sie die Anwendung? Und von wo?
- Besteht ein Zusammenhang zwischen externen Ereignissen und Aktivitätsspitzen?
- Bezieht sich der Business-Plan des Unternehmens auf die Aktivität meiner Anwendung?
- Wird sich die User Community in verschiedene geografische Gebiete ausdehnen?
- Ist ein Marketingplan zur Vermarktung beziehungsweise Förderung der Anwendung vorhanden? Wenn ja, wer ist die Zielgruppe?
- Werden manche Ebenen der Architektur mit (einem) anderen System(en) gemeinsam genutzt?

Die Idee ist es, die Anwendung, die Gewohnheiten der Endkunden und die Beziehung zwischen der Anwendung und der aktuellen oder zukünftigen Organisation zu verstehen.

Ein wichtiges Element des Performance-Designs ist die Denkzeit („Think Time“). Es ist die Zeitspanne, die ein echter Benutzer zwischen zwei Geschäftsaktionen benötigt:

- Zeit zum Lesen der auf dem Bildschirm angezeigten Informationen sowie
- Zeit zum Ausfüllen der Informationen in einem Formular und
- andere Aktionen der echten Benutzer, die keine Interaktion mit den Anwendungsservern verursachen.

Da sich jeder echte Benutzer unterschiedlich verhält, werden die Denkzeiten stets verschieden sein.

Ein SLA ermöglicht es dem Performance-Techniker, einen Status über die Ergebnisse der Performance-Tests zu liefern. Mit dem SLA können Performance-Tests einfach automatisiert werden, um die Performance-Regression zwischen verschiedenen Versionen der Anwendung zu identifizieren.

Die meisten Projekte konzentrieren sich nur darauf, die Grenze der Anwendung zu erreichen. Das Erreichen der Anwendungsgrenze ist wichtig, um die Dimensionierung und die Konfiguration der Architektur zu überprüfen, aber dies wird nicht allen Leistungsanforderungen entsprechen.

Nachstehende Tests werden bei der Überprüfung der Leistungsanforderungen helfen:

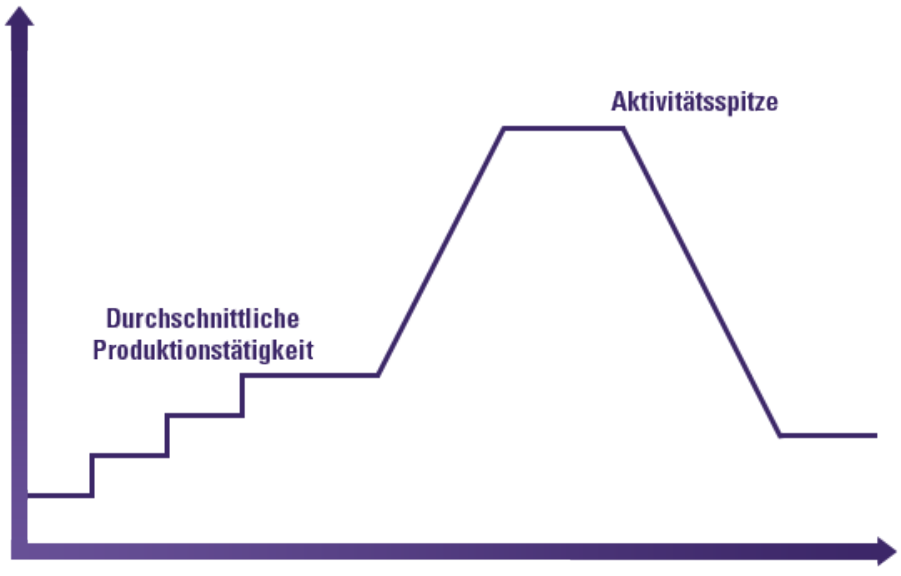

Abb. 2: Modell eines Peak-Tests

- *Modultest:* Viele Projekte übersehen die Macht von Modultests. Sie sollten keine Lasttests starten, wenn die Performance mit einem einzigen Benutzer nicht akzeptabel ist.
- *Verbindungstest:* Nehmen wir an, dass alle Anwendungsbenutzer jeden Morgen etwa zur gleichen Zeit eine Verbindung zum System herstellen und sich dann einen Kaffee holen oder mit ihren Kollegen plaudern. Auch wenn keine größeren Aktivitäten an den Business-Komponenten der Anwendung stattfinden, muss die Architektur diese sehr hohe Spitze von Benutzersitzungen bewältigen.
- *Produktivbetriebstest:* Jede Anwendung weist bedeutende Geschäftsaktionen auf. Diese Aktionen führen oft dazu, dass Daten aktualisiert oder in die Datenbank eingefügt werden. Der Produktivbetriebstest stellt sicher, dass das System in der Lage ist, die Last in Bezug auf die in der Produktion erwartete Anzahl der Transaktionen pro Stunde zu bewältigen.
- *Peak-Test:* Jede Anwendung erfährt Volumenspitzen. Auch wenn solche Spitzen nur ein- oder zweimal pro Jahr auftreten, muss unbedingt sichergestellt werden, dass die Anwendung diese bewältigen kann (siehe Abbildung 2).
- *Soak-Test:* Wenn keine speziellen Zeiten für Wartungsarbeiten vorgesehen sind, dann ist es wichtig, dass die Architektur fehlerfrei über einen längeren Zeitraum laufen kann.
- *Batch-Test:* Manche Anwendungen werden mit asynchronen Aufgaben oder Batches entworfen. Welche Auswirkungen hat ein Batch auf die echten Benutzer?

Je nach den Einschränkungen des Geschäfts und den Standorten der Benutzer könnte eine Reihe anderer Lasttests eingesetzt werden. Es finden immer wieder Ereignisse im Zusammenhang mit dem Unternehmen oder der Organisation statt, die die Last der Anwendung beeinflussen können.

So wird zum Beispiel die für eine Geschäftsanwendung bemessene Last von den Öffnungszeiten der verschiedenen Märkte abhängen. Oftmals wird E-Business durch großvolumige Marketingaktivitäten begleitet, was ebenfalls zu sehr unterschiedlichen Lastprofilen führt.

Auf der anderen Seite werden Tests entwickelt, um die Verfügbarkeit der Plattform während der Wartungsarbeiten oder einer Produktionsstörung zu überprüfen:

- *Failover-Test:* Ziel dieses Tests ist es, den Einfluss eines Produktionsausfalles auf die Umgebung zu simulieren. Diese Art von Test ist zwingend notwendig, um die Verfügbarkeit der Umgebung zu überprüfen und sicherzustellen, dass der Failover-Cluster-Mechanismus richtig reagiert und zwar über alle Schichten der Architektur: Webserver, Anwendungsserver, Datenbank usw. Es sollte ein Test pro Schicht erfolgen.
- *Recovery-Test:* Der Test wird durch Stoppen und anschließendes Neustarten eines der Knoten gestartet, während die Anwendung geladen wird. Mit diesem Test soll erkannt werden, wie der Knoten gleich nach dem Neustart auf eine hohe Auslastung reagiert.

Es gibt noch einen weiteren Punkt, der die Performance der Anwendung beeinflussen könnte: die Daten. Datenbestände wachsen in der Produktion recht schnell. Je nach Größe der Datenbank ist ihr Verhalten sehr unterschiedlich.

Es ist zu überprüfen, ob die Grenze zwischen einer kleinen und einer großen Datenbank unterschiedlich ist. Es darf niemals eine Liste von Namen verwendet werden, in der alle Posten mit AA..., AAA2 ... beginnen, sondern stattdessen ein repräsentativer Datensatz, der von A auf Z zeigt.

Art und Komplexität des Tests werden sich während des Projektlebenszyklus ändern, ebenso die benötigten Testumgebungen (siehe Abbildung 3).

Die Benutzerfreundlichkeit überprüfen

Mobile Anwendungen, Rich Internet Applications (RIA) und komplexe Ajax-Frameworks stellen in der Art und Weise, wie die meisten zur Messung der Antwortzeiten verwendet werden, eine Herausforderung dar. In der Vergangenheit wurden Messungen auf die Download-Zeit und die Zeit bis zum ersten Byte (TTFB) beschränkt.

Dieser Ansatz ist nicht sinnvoll, da ein Großteil der Renderzeit, die zur Benutzerfreundlichkeit beiträgt, vom lokalen ActiveX/JavaScript oder von der nativen Anwendungslogik abhängt.

Das Lasttest-Tool wird 98 Prozent der Last an der Anwendung erzeugen. Das browserbasierte oder mobile Testtool wird die restlichen 2 Prozent der Last erzeugen, um die echte Benutzererfahrung (einschließlich der Renderzeit) zu erhalten, während die Anwendung geladen wird.

Überwachung

Das Ausführen von Tests ohne Überwachung ist wie das Ansehen eines Horrorfilms im Radio. Sie werden Leute schreien hören, ohne zu wissen warum. Die Überwachung ist die einzige Möglichkeit, Messwerte über das Verhalten der Architektur zu erhalten.

Mit der Überwachung kann die Architektur besser verstanden und das Verhalten der verschiedenen Teile der Umgebung untersucht werden:

- Betriebssystem: CPU, Speicher, Festplatte, Netzauslastung usw.

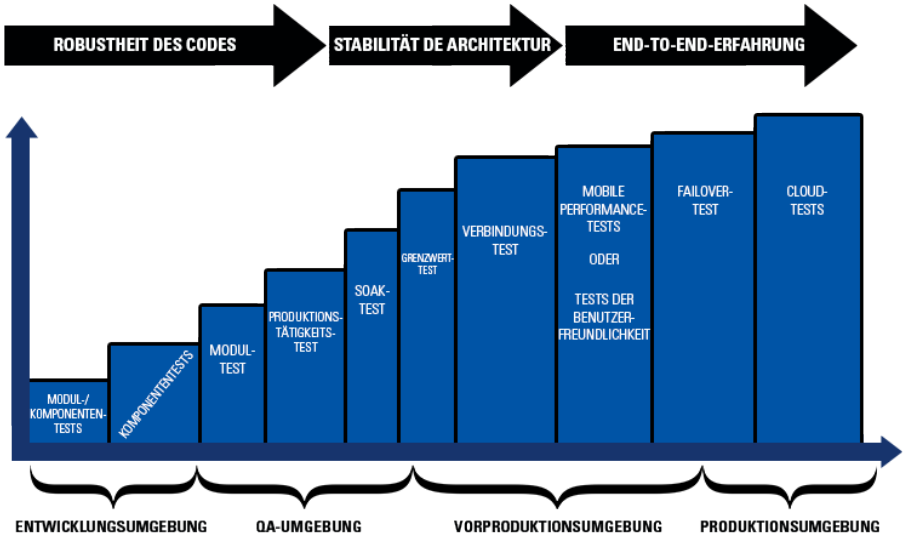

Abb. 3: Zuordnung der Testtypen zu den Testumgebungen

- Anwendungsserver: Speichernutzung, Thread-Nutzung, Sitzungen usw.
- Webserver: Arbeitskraft, Anzahl der Sitzungen usw.
- Datenbank: Pufferpool, Cache-Nutzung, Anzahl der Transaktionen, Anzahl der Commits, Prozentsatz der indexierten Anfragen usw.
- Caching-Server: Trefferquote

Viele Projekte verwenden Produktionsüberwachungstools, um Messdaten über die Architektur zu erhalten. Diese Vorgehensweise ist nicht empfehlenswert, da die Produktionsüberwachung eine zu große Granularität zwischen jedem Datenpunkt (alle 2-5 Minuten) aufweist. In Lasttests ist es wichtig, dass überwachte Daten mindestens alle fünf Sekunden erfasst werden. Wenn während eines Tests eine Lastspitze nur für wenige Sekunden auftritt, ist es wichtig, über genug Granularität zu verfügen, um den Engpass aufzeigen zu können.

Ausführung eines Performance-Tests

Das Testen der Performance ist ein iterativer Prozess mit den folgenden vier Phasen (siehe Abbildung 4).

Design

Die Design-Phase ist oft als „Scripting"-Phase bekannt. Die neuen Technologien haben die Design-Phase jedoch sehr komplex gemacht.

Das Erstellen von Performance-Testskripts stellt an sich schon ein Softwareentwicklungsprojekt dar. Manchmal wird eine automatische Skriptgenerierung von der Aufzeichnung fälschlicherweise als der gesamte Prozess der Skripterstellung interpretiert, aber es ist nur der Anfang: Skripte müssen korreliert (dynamische Variablen vom Server erhalten) und parametriert werden (unterschiedliche Daten für verschiedene Benutzer verwenden). Ajax-Anwendungen, WebSocket und Abfragetechnologien erzeugen regelmäßig Anfragen. Diese Anfragen werden nicht durch die Interaktion des Benutzers innerhalb der grafischen Benutzerschnittstelle erzeugt. Stattdessen werden viele Anrufe durch eine oder mehrere einzelne Seitenkomponenten erzeugt. So findet man zum Beispiel auf einer E-Commerce-Website immer einen direkten Link von den Seiten zum Warenkorb des Benutzers. Die meiste Zeit wird die Anzahl der Produkte über eine Abfrageanforderung angezeigt (ein technischer Anruf alle fünf Sekunden). Diese Art von Anruf muss vom Performance-Techniker verstanden werden.

Ein weiteres gutes Beispiel ist die adaptive Streaming-Technologie. Viele Tester verlassen sich oft auf den Aufnahme-/Wiedergabe-Ansatz. Leider wird diese Art von Logik nicht jeder Teststrategie nützen. Ziel ist es, das Verhalten der Anwendung unter realistischer Last zu qualifizieren, nicht die Überprüfung der Zwischenspeicherung. Bei einer erneuten Wiedergabe der Aufzeichnungsanforderung ohne Korrelation wird nur der Zwischenspeicher der Anwendung aufgerufen, nämlich der Web- und der Anwendungs-Cache.

Ein weiteres wichtiges Element des Design-Schrittes ist der Datensatz. Mit einem kleinen Datensatz wird genau dieselbe Abfrage an die Datenbank gestellt. Wie bereits erwähnt, geht es bei einem Lasttest nicht darum, die Effizienz des Datenbank-Cache zu qualifizieren oder einfach Deadlocks zu erzeugen.

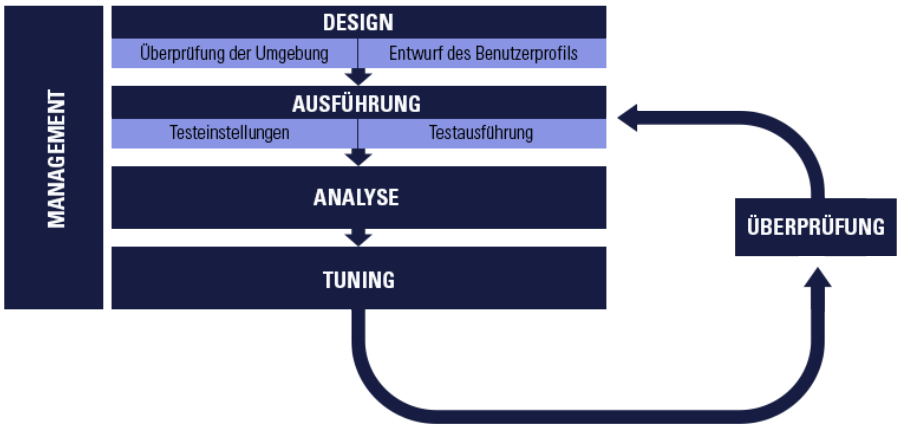

Abb. 4: Performance Testen ist ein iterativer Prozess

Ausführung

Die meisten Lasttest-Lösungen bestehen aus mehreren Komponenten:

- Controller: Dieser steuert den Test und speichert die Ergebnisdaten: Antwortzeit, Treffer/s, Datendurchsatz, Überwachungskennzahlen, Fehler usw.
- Lastgenerator: Dieser führt das Lasttest-Skript gegenüber der Anwendung aus. Diese Komponente muss mehrere hundert gleichzeitige virtuelle Benutzer bewältigen.

Um Einschränkungen durch die Lasttest-Architektur zu vermeiden, müssen nachfolgende Elemente in ausreichender Menge vorhanden sein:

- Lastgeneratoren zur Erzielung der erwarteten Last,
- Netzwerk-Bandbreite zwischen Lastgenerator und Anwendung

Die erste maßgebliche Belastungsgrenze, die sich aus den Kennzahlen hinsichtlich CPU, Speicher, Datendurchsatz oder Trefferquote ergibt, stellt die Leistungsgrenze des Lastgenerators dar. Dieser Punkt muss mit der Anzahl der generierten virtuellen Benutzer abgeglichen werden. Ernste Probleme mit dem Lastgenerator können dann auftreten, wenn dieser über die jeweilige Anzahl der virtuellen Benutzer hinaus belastet wird.

Effiziente Testüberwachung

Wenn die Antwortzeiten aufgrund der maximalen Auslastung eines Webservers zunehmen, sollten Sie den Test beenden und den Webserver entsprechend abstimmen. Bei nahezu jedem Testlauf auf einer neuen Anwendung und/oder neuen Umgebung sind die meisten Schichten (Anwendungsserver, Webserver usw.) für die spezifische Last der Anwendung nicht konfiguriert oder abgestimmt. Daher erfordert jeder Test auf einer repräsentativen Umgebung große Aufmerksamkeit des Performance-Technikers, um die Umgebung ordnungsgemäß abzustimmen.

Wenn Sie jedoch umfassende Kenntnisse über die Anwendung und Umgebung besitzen, kann die Testautomatisierung auch gestartet werden, ohne das Verhalten der Anwendung zu überwachen. Nach Beendigung des Tests müssen Sie die Ergebnisse analysieren.

Analyse

Die Analyse der Ergebnisse eines Lasttests stellt an sich schon ein großes Stück Ar-

beit dar. Sie erfordert umfassende Kenntnisse über die nachfolgenden Elemente:

- das Design des Lasttests,
- die in der Anwendung enthaltenen technischen Schichten und
- die moderne Architektur.

Nahezu alle Lasttest-Lösungen ermöglichen die Erstellung komplexer Diagramme zur Abgleichung der Daten. Vor der Erstellung dieses Berichts ist es jedoch unerlässlich, die Funktion und das technische Know-how der Person zu verstehen, die den Bericht prüft oder einfach nur durchliest.

Der Hauptzweck eines Berichts über den Performance-Test besteht darin, einen anschaulichen Status über die Performance der Anwendung bereitzustellen. Ergebnisberichte sollten einfach strukturiert sein und sich auf diese drei Anwendungsgebiete konzentrieren: Antwortzeiten, Verfügbarkeit und Skalierbarkeit.

Eine grafische Darstellung (z. B. mit Kreisdiagrammen) erhöht die Verständlichkeit der Ergebnisse für Entscheidungsträger.

Abschließend sollte der Performance-Bericht noch hervorheben, ob die Leistungsanforderungen überprüft wurden.

Fazit

Für den Erfolg eines IT-Projektes ist die Performance wichtiger denn je. Um den größtmöglichen Nutzen aus Last- und Performancetests zu ziehen, ist dringend zu empfehlen, die Testmethodik an die Entwicklungsmethodik anzupassen.

Gleichzeitig muss die Durchführung von Lasttests, als fester und unverrückbarer Bestandteil, in die Planung und Durchführung von Software-Projekten Eingang finden. ■

Kontakt für weitere Informationen

USA: Tel.: +1 781 899 7200
EMEA: Tel.: +33 442 180 830
DACH: Tel.: +49 89 5130 2264
E-Mail: sales@neotys.com
Weitere Informationen: www.neotys.de

Links

[Coll04] R. Collard, DETERMINING THE TEST FOCUS THROUGH RISK ASSESSMENT (ERMITTLUNG VON TESTSCHWERPUNKTEN MITTELS RISIKOBEWERTUNG), siehe http://www.performance-workshop.org/documents/Determining_Test_Focus_Thru_Risk_assessment_Collard.pdf