Las(s)t die Tests beginnen!

Ingo Rauschenberg | ingo.rauschenberg@cms.hu-berlin.de
Johannes Caspary | johannes.caspary@cms.hu-berlin.de

Das Warten auf den Aufbau einer Webseite wird sicher jeder, der das Internet nutzt, schon einmal erlebt haben. Heutzutage ist es einfacher geworden, Webanwendungen zu schreiben und zu betreiben. Bei der Entwicklung achtet man vor allem auf Funktionalität und Design. Was ist aber, wenn nicht nur einzelne Nutzer eine solche Webseite aufrufen, sondern Hunderte oder Tausende gleichzeitig auf die Anwendung zugreifen? Wie verhält sich das System also unter Last?
JMeter ist ein Werkzeug, das Administratoren hilft, einen Online-Dienst einer künstlichen Last auszusetzen, um Probleme beim Aufbau zu erkennen. Der Artikel gibt Einblicke, wie Lasttests mit JMeter durchgeführt werden können und welche Schlüsse man aus ihren Ergebnissen ziehen kann.

Was sind Lasttests?

Wenn man einen Beschäftigten der IT-Branche danach fragt, was er unter Lasttests versteht, so wird man wahrscheinlich die wenig überraschende Antwort bekommen, dass Lasttests spezielle Tests sind, die Lastprobleme aufdecken sollen. Es stellt sich nun also die Frage, was sich hinter diesem Begriff verbirgt. Um nicht gleich mit der IT-Sprache zu antworten, sollen zwei kleine Beispiele Lastprobleme verdeutlichen.

Stellen Sie sich vor, Sie gehen in den Supermarkt einkaufen und bezahlen an der Kasse. Nun kommt der nächste Kunde zu dieser Kasse, um zu bezahlen. Wenn immer mehr Kunden diese Kasse wählen und dies schneller geschieht, als der Kassierer die eintreffenden Kunden bedienen kann, entsteht eine Schlange – und diese Kasse hat ein Lastproblem. Sie wird mit der gerade anstehenden Last, den Kunden, nicht sofort fertig. Die Lösung dieses Problems sollte Ihnen bekannt vorkommen: Der Kassierer bemerkt die länger werdende Schlange und ruft über die Lautsprecher einen Kollegen zu Hilfe. In diesem ersten Beispiel ist das Erkennen und Lösen des Lastproblems recht einfach.

Im zweiten Beispiel nehmen wir einmal an, dass Sie, bevor Sie aus dem Haus gehen, noch schnell ein paar Musikstücke von Ihrem Rechner auf den MP3-Player kopieren wollen. Sie wählen die Songs aus und ziehen sie auf den Player. Dann erscheint der Dialog, der Ihnen mitteilt, dass es noch zehn Minuten dauert, bis das Kopieren fertig ist. Haben Sie es nun eilig, so wird aus dem Kopiervorgang für Sie ein Lastproblem. Hier ist das Erkennen der Ursache des Lastproblems schon schwieriger. Zum Beispiel kann Ihr Rechner einfach eine zu langsame Festplatte haben und so die Daten nicht schnell genug liefern. Es ist aber auch möglich, dass Ihr Rechner und der MP3-Player der Meinung sind, Daten mittels USB 1.0 und nicht mit USB 2.0 auszutauschen, was ca. 40 mal langsamer ist. Daneben gibt es auch weitere Möglichkeiten, warum das einfache Kopieren so viel Zeit beansprucht. Die Beseitigung einer Ursache kann zwar den größten Flaschenhals im Datenstrom beseitigen, aber der zweitgrößte wird unmittelbar nach deren Beseitigung spürbar.

Für derartige Fälle sind in der IT-Branche Lasttests vorgesehen. Ziel ist es, alle Komponenten, die unter Lastbedingungen ein Problem verursachen, zu erkennen. Anschließend wird analysiert, was getan werden kann, um dieses zu beseitigen.

Wenn man die Humboldt-Universität zu Berlin betrachtet, gibt es auch hier IT-Systeme, die zu bestimmten Terminen großer Last standhalten müssen. Als Beispiel soll das System AGNES – Lehre und Prüfung Online dienen. In AGNES findet jedes Semester sowohl die Veröffentlichung des Vorlesungsverzeichnisses des kommenden Semesters als auch die Belegung von Veranstaltungen statt. An diesen Terminen wird das System von sehr vielen Studierenden, Lehrenden und auch Gästen genutzt. Ein Lastproblem zu diesen Zeitpunkten ist zum einen für die Nutzer ärgerlich, zum anderen aber auch für das Image der Humboldt-Universität bzw. des CMS als Dienstbetreiber schädlich.

Abbildung 1 zeigt vereinfacht den Aufbau der beteiligten IT-Komponenten von AGNES. Zu sehen ist, dass eine

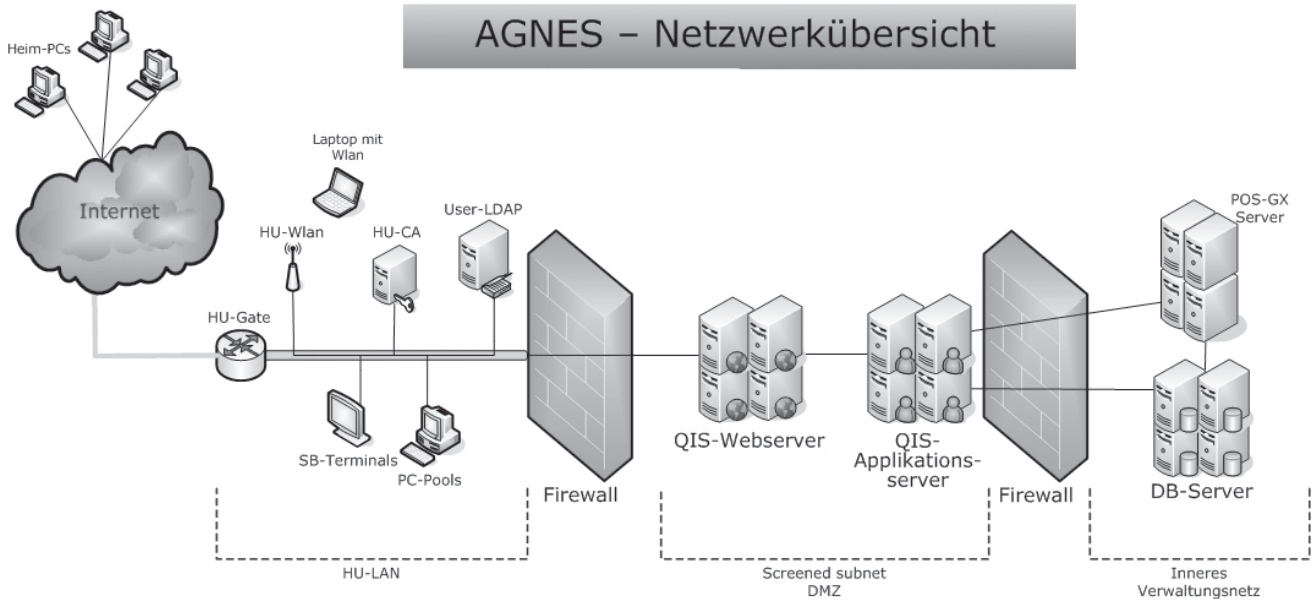

Abb. 1: AGNES Aufbau der IT-Komponenten

äußere und eine innere Firewall eine Rolle spielen und dass neben Webservern auch Tomcat-Applikationsserver und ein PostgreSQL-Datenbankserver für den Betrieb von AGNES benötigt werden. Wenn also bei AGNES ein Lastproblem auftritt, können die Gründe dafür an vielen verschiedenen Stellen sitzen. Es ist keine gute Idee, sich der Schwachstellen erst dann anzunehmen, wenn die Lastprobleme akut werden. Denn eventuell löst man nur eines der Probleme, während sich danach an einer ganz anderen Stelle im System ein Neues auftut. Ziel von Lasttests ist es daher, vor dem realen Auftreten dieser Probleme mit einer künstlich erzeugten Last alle kritischen Punkte zu identifizieren (vergleiche Abbildung 2). Um eine solche Last zu erzeugen, gibt es verschiedene Methoden, von denen die Einfachste eine Gruppe Freiwilliger ist, die das IT-System nutzen. Da jedoch nicht immer die nötige Anzahl freiwilliger Tester zur Hand ist und die Automatisierung Informatikern im Blut liegt, gibt es Programme, deren Funktion die Erzeugung solcher Lasten ist. In Kasten 1 sind verschiedene dieser Projekte kurz beschrieben. Im Folgenden soll jedoch auf das Programm JMeter der Apache Software Foundation [1] näher eingegangen werden.

Alternativen zu JMeter

JMeter ist ein leistungsstarkes und verbreitetes Werkzeug zur Durchführung von Lasttests. Allerdings gibt es zahlreiche Alternativen, die hier kurz vorgestellt werden. Der Fokus liegt auf Open-Source-Software.

Wie auch JMeter ist die Mehrzahl der frei verfügbaren Lasttestwerkzeuge in Java geschrieben. Damit sind sie grundsätzlich auf allen Plattformen lauffähig. Strukturell ähnelt vor allem The Grinder [2] dem Ansatz von JMeter, allerdings werden die Testfälle bei The Grinder statt in XML in Jython[1] formuliert. Die Lizenz lehnt sich an der BSD Open-Source-Lizenz an und gestattet die Verwendung und Modifizierung der Quellen [3].

Ursprünglich nur für funktionale Tests auf Web-Services entwickelt, unterstützt soapUI [4] von Eviware Software auch Tests auf Webseiten (über HTTP/HTTPS) und Datenbanken (über JDBC). Neben der über SourceForge frei verfügbaren Version wird auch eine kommerzielle Version angeboten. Die Ausführung der Lasttests erfolgt über loadUI [5], das soapUI integriert. Die verteilte Ausführung erfolgt über sogenannte loadUI-Agents.

1 Jython ist eine Portierung von Python in die Java VM.

Speziell für Datenbanklasttests wurde Bristlecone [6] entwickelt. Die üblichen SQL-Befehle für den Test können automatisch generiert werden. Die Auswertung der Daten ist über HTML, XML oder CSV möglich – eine grafische Unterstützung ist noch nicht vorhanden. Es ist unter GNU General Public License (GNU GPL) Version 2.0 lizenziert und steht über SourceForge zum Download bereit. Dieses Tool wurde für die Tests der Datenbankserver innerhalb der Abteilung "DV in der Verwaltung" des CMS eingesetzt.

Nicht alle Open-Source-Werkzeuge für Lasttests sind auf allen Plattformen lauffähig. So kann OpenSTA [7], das in C++ programmiert wurde, lediglich auf Windows-Maschinen betrieben werden. Eingesetzt wird OpenSTA für Lasttests von Webanwendungen. Die HTTP/HTTPS-Anfragen werden über einen Proxy aufgezeichnet. Für die verteilte Ausführung von OpenSTA wird CORBA[2] verwendet, die Testformulierung erfolgt in der proprietären Skriptsprache SCL (Script Control Language). Die Quellen werden unter SourceForge gehostet und stehen unter GNU GPL in Version 2.0.

2 CORBA = Common Object Request Broker Architecture; objektorientierte Infrastruktur für verteilte Anwendungen, siehe auch [8]

Neben frei verfügbaren Programmen zur Durchführung von Lasttests gibt es auch zahlreiche kommerzielle Varianten. Sie zeichnen sich im Allgemeinen durch die Unterstützung von selteneren Protokollen, einer komfortableren Bedienoberfläche und umfangreiches Reporting aus.[3] Weiterhin wird u.a. die Ausführung der Tests über den Anbieter selbst ermöglicht (via Cloud Computing [10]). Allerdings sind kommerzielle Lasttestwerkzeuge mitunter sehr kostspielig.[4] Bekannte kommerzielle Anbieter sind z. B. Neotys [11] und Compuware [12].

Bei den bisher genannten Werkzeugen handelt es sich um mehr oder weniger vollständige und integrierte Programme zur Durchführung von reproduzierbaren Lasttests.

In bestimmten Situationen, wie zur schnellen Untersuchung von Lastproblemen im laufenden Betrieb, bietet sich das kommandozeilenbasierte Tool Apache Benchmark [13], oder kurz *ab*, an. Ursprünglich wird *ab* für das Benchmarking des HTTP-Servers von Apache eingesetzt. Durch einen kurzen Befehl kann eine Anzahl von nebenläufigen Requests auf eine Webseite ausgeführt werden (z. B. *ab -n 100 -c 2 http://agnes.hu-berlin.de/* für 100 Requests, von denen jeweils zwei gleichzeitig ausgeführt werden).

Obwohl es kein richtiges Lasttestwerkzeug ist, soll an dieser Stelle auch Selenium [14] erwähnt werden. Mit Selenium werden üblicherweise funktionale GUI-Tests auf Webanwendungen ausgeführt. Es ist in Javascript implementiert und lässt sich sowohl über ein Firefox Plugin (Selenium IDE) als auch direkt über Selenium-Bibliotheken in vielen Programmiersprachen ausführen. Selenium kann bei der Planung für die Aufzeichnung von Lasttests genutzt werden. Eine Anwendung zur gehosteten Integration von Selenium zur Ausführung von Lasttests ist browsermob [15].

3 siehe [9], Punkt 3.3.1
4 siehe [9], Punkt 3.3.2

Kasten 1: Liste einer Auswahl verschiedener Lasttestprogramme neben JMeter

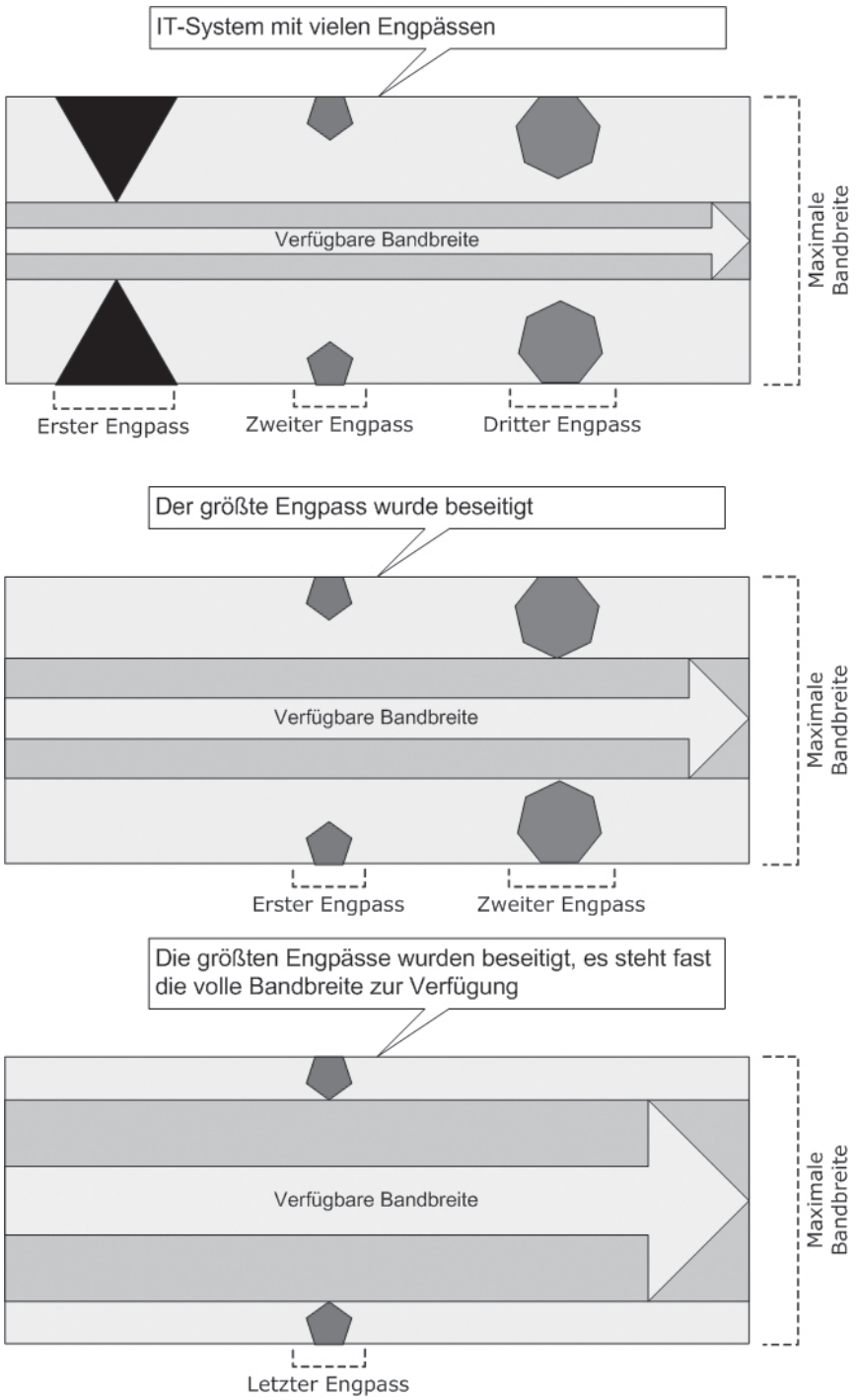

Abb. 2: die Beeinflussung der verfügbaren Bandbreite durch Engpässe

Apache JMeter

Zur Analyse der eingangs erläuterten Lastprobleme in der IT bietet sich Apache JMeter an. Es ist ein freies, in Java geschriebenes Tool, das im Rahmen des Jakarta-Projektes der Apache Software Foundation entwickelt wird. JMeter bietet eine grafische Oberfläche, in der Testpläne erstellt und ausgeführt werden können.

Mit JMeter ist es möglich, verschiedene Anwendungen zu testen. Dazu gehören z. B. Mailserver, LDAP[5]-Server,

5 Lightweight Directory Access Protocol

Datenbankserver via JDBC[6], FTP[7]-Server, SOAP[8]-Schnittstellen und auch Webserver. Hierfür werden in JMeter Testpläne erstellt, in denen die entsprechenden Anfragen, in JMeter Sampler genannt, für die Server definiert werden. Diesen Samplern können verschiedene Parameter mitgegeben werden. Neben den Samplern, die die Anfragen an die Server senden, hat JMeter die Möglichkeit, innerhalb der Testpläne mit Logik-Elementen, Zeitgebern und Konfigurations-Elementen den Testablauf zu beeinflussen. So kann für einen Test, der mehrere Abfragen an einen Webserver enthält und der eine Nutzer-Session[9] abbilden soll, beispielsweise ein Cookie[10]-Container eingefügt werden. So bleibt für die verschiedenen Sampler das Session-Cookie erhalten. Daneben ist es auch möglich, Werte für Variablen, die in Samplern genutzt werden, als Pool zu hinterlegen und diese über eine Zufallsfunktion zu nutzen.

Außerdem bietet JMeter zur Erstellung der Testpläne verschiedene Ausgabeoptionen, sogenannte Listener, an. Mit diesen Listenern kann man wahlweise die Ergebnisse in Dateien speichern sowie sich verschiedene aggregierte Statistiken über die Antworten oder über die mehrfache Wiederholung des Testplans anzeigen lassen.

Um die eben erwähnte Mehrfachausführung eines Testplans zu realisieren, werden die beschriebenen Elemente des Plans zu sogenannten Threadgruppen zusammengefasst. Für diese Gruppen wird in JMeter definiert, wie viele Durchläufe parallel zu starten und wie oft diese zu wiederholen sind.

Abbildung 3 zeigt einen Testplan für die AGNES-Seite. Es wird jeweils in einer eigenen Threadgruppe die Startseite von AGNES aufgerufen, einmal via HTTP und parallel dazu via HTTPS. Die Ergebnisse der einzelnen Threadgruppen werden aggregiert ausgegeben. Zusätzlich

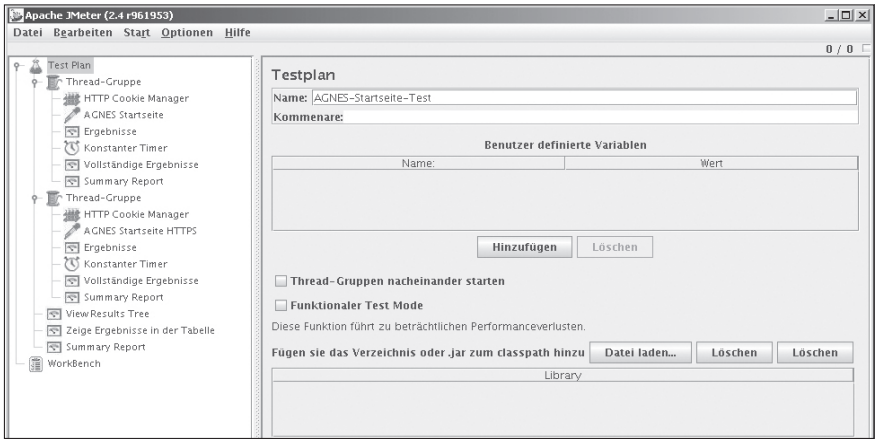

Abb. 3: ein Testplan für die AGNES-Startseite

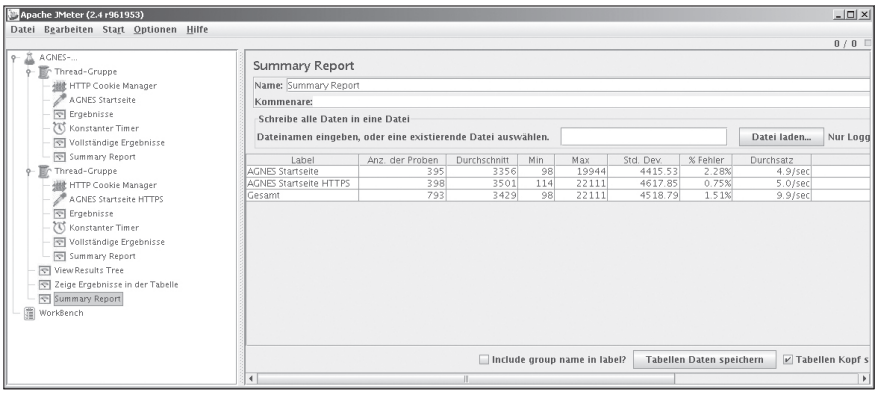

Abb. 4: Ansicht aggregierter Ergebnisse

erfolgt eine gesammelte Ausgabe über beide Threadgruppen zusammen. In Abbildung 4 ist eine Zusammenfassung der Ergebnisse der beiden Threadgruppen zu sehen, nachdem diese eine Weile gelaufen sind.

Da das Erstellen von Testplänen, insbesondere im Webserver-Bereich, recht kompliziert und langwierig sein kann, bietet JMeter eine Möglichkeit der Vereinfachung. JMeter lässt sich so konfigurieren, dass es als Proxyserver für den eigenen Webbrowser dient. Es zeichnet im Folgenden alle Aktionen des Browsers auf und fügt diese einem vorher leeren Testplan hinzu. Durch diese Funktionalität kann man zuerst selbst den gewünschten Testverlauf im Browser durchspielen und sich dann den daraus resultierenden Testplan ansehen und verfeinern. Ein Nachteil hiervon ist allerdings die Beschränkung auf das HTTP-Protokoll. Will man Webseiten testen, die nur über eine verschlüsselte Verbindung via HTTPS erreichbar sind, funktioniert es auf diesem Weg nicht. Hier muss man zumindest für den Testlauf, bei dem der JMeter-Proxy aufzeichnen soll, von Serverseite den Zwang zur Verschlüsselung der Webseite deaktivieren.

Ein JMeter-Lasttest kann auch auf dem Rechner, der den Test ausführen soll, eine sehr hohe Last erzeugen. Man stelle sich einen Test vor, der einen Ansturm von mehr als 5000 Studierenden gleichzeitig mit den entsprechenden parallel laufenden Threads umsetzen soll. Durchschnittliche und selbst besser ausgestattete Arbeitsplatzrechner sind für diese Ausführung nicht mehr geeignet.

Um dem Problem zu begegnen, kann JMeter neben der Ausführung über die grafische Oberfläche auch im Servermodus ausgeführt werden. In diesem Modus nimmt JMeter Testpläne via RMI[11]-Aufruf zur Ausführung entgegen und liefert die Ergebnisse an einen zentralen Rechner, den Master. Der übernimmt dann die Darstellung und Auswertung der so eingesammelten Ergebnisse von

6 Java Database Connectivity
7 File Transfer Protocol
8 SOAP stand ursprünglich für Simple Object Access Protocol, wird jedoch seit Version 1.2 nicht mehr als Akronym verwendet.
9 Als Session bezeichnet man eine stehende Verbindung eines Clients mit einem Server.
10 Mithilfe eines Cookies kann ein Webserver Daten auf einem Clientrechner speichern. So kann beispielsweise bei dem zustandslosen Protokoll HTTP eine Session realisiert werden.

11 Remote Method Invocation

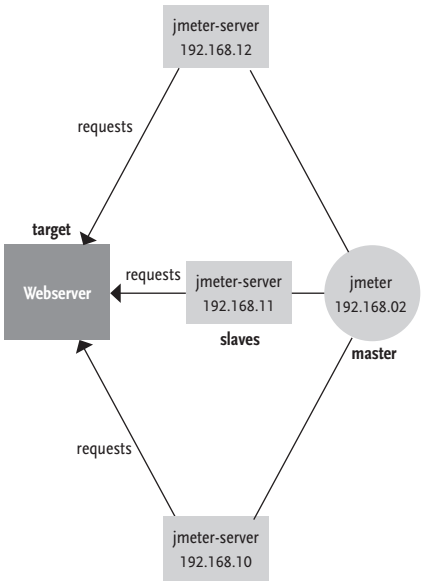

Abb. 5: Verteilte Lasttests [16]

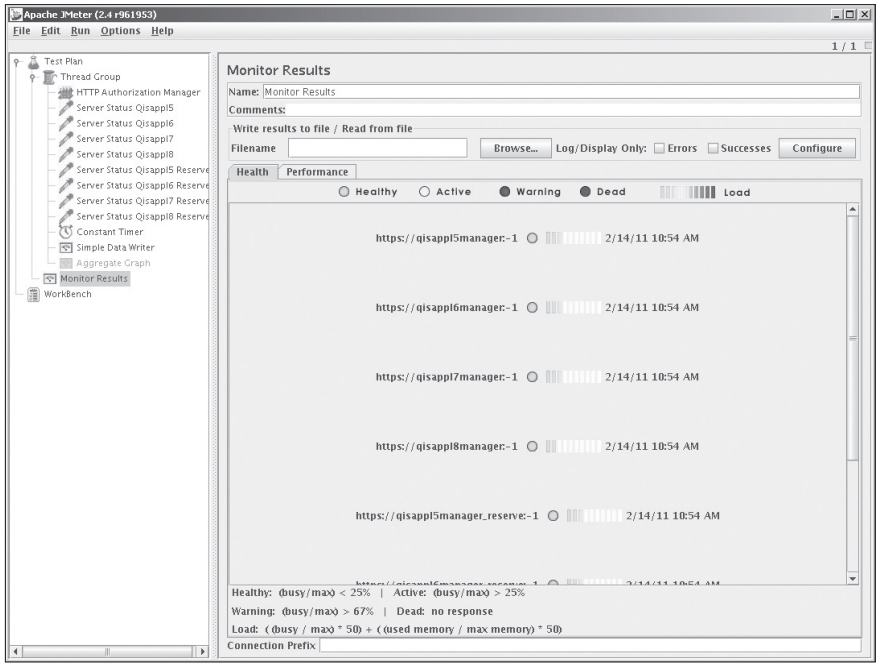

Abb. 6: Tomcatüberwachung – Health-Monitor

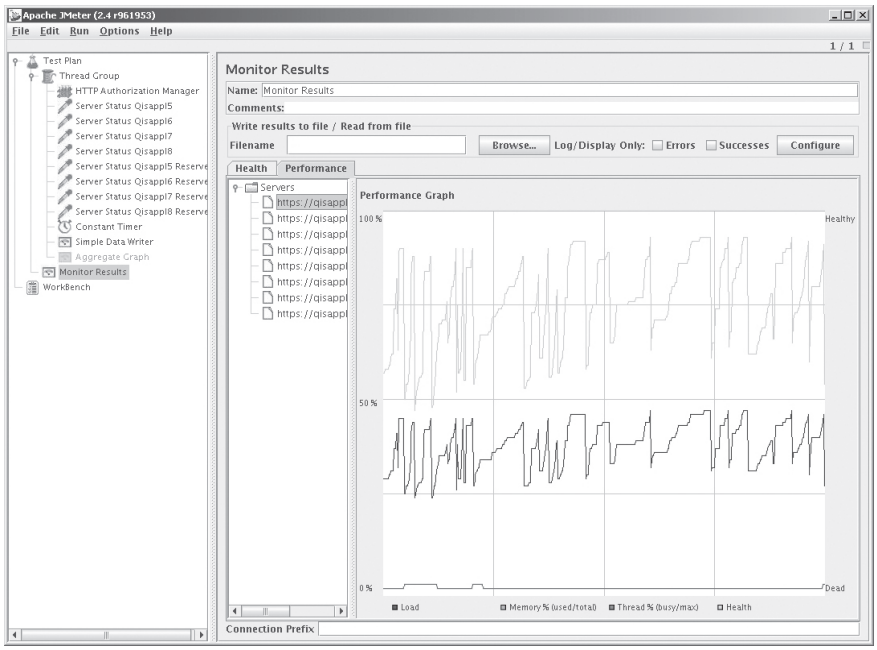

Abb. 7: Tomcatüberwachung – Performance-Monitor

verschiedenen JMeter-Servern in den Listenern. Abbildung 5 zeigt, wie man sich einen so verteilten Lasttest vorstellen kann. Der Master ist der Rechner, auf dem JMeter als grafische Oberfläche ausgeführt wird. Von dort wird der Testplan an die drei JMeter-Server, die Slaves, verteilt. Diese führen den Testplan, der Anfragen an den Webserver sendet, aus und melden die Ergebnisse zurück an den Master.

In der Abteilung „DV in der Verwaltung“ des CMS existiert ein solches JMeter-Server-Cluster. Mit diesem werden die Lasttests des für Studium und Lehre kritischen Systems AGNES und auch anderer Systeme, durchgeführt.

JMeter zum Monitoring von Tomcat-Servern

Ein Vorteil von JMeter, der sich aus der Entwicklung als Teil des Jakarta-Projektes ergibt, ist die Möglichkeit, mithilfe von JMeter Tomcat-Server zu überwachen und deren Auslastung grafisch darzustellen. Hierfür wird der im Tomcat-Manager verfügbare XML-Status-Report als Sampler in den Testplan eingefügt. Für die Threadgruppe, die diesen Sampler enthält, wird eine Ergebnisüberwachung als Listener eingefügt. Die XML-Tomcat-Statusreports werden von JMeter in speziell aufbereiteter Form dargestellt. In Abbildung 6 ist eine solche Überwachung für acht Tomcat-Server von AGNES, die auf vier physische Server verteilt sind, dargestellt. Die Health-Zusammenfassung gibt eine kurze Übersicht und in der in Abbildung 7 dargestellten Performanceansicht werden die Speicherauslastung, die Anzahl laufender Threads sowie die Last des Servers über einer Zeitachse angezeigt.

Nach dem Lasttest

Auch wenn man mit JMeter oder einem anderen Tool eine Software besitzt, mit der man Lasttests durchführen kann, so ist mit deren Erstellung und Durchführung nur die halbe Arbeit getan. Neben der möglichen Erkenntnis, dass es ein Lastproblem gibt, muss bei einem System mit mehreren beteiligten Komponenten auch ausfindig gemacht werden,

an welcher Stelle das Lastproblem auftritt. Hier bietet der Lasttest die Möglichkeit, den Problemfall quasi auf Knopfdruck wiederherzustellen. Wenn dieser auftritt, können die beteiligten Komponenten von den Administratoren durch Analyse der Logdateien, System-Monitoring oder andere Verfahren genauer untersucht werden. Sobald für ein bestehendes Lastproblem eine Lösung gefunden wird, muss der Test natürlich wiederholt werden. Wie in Abbildung 2 dargestellt, kann es sein, dass das gefundene Problem nicht das Einzige war. Und dann geht die Suche weiter.

Ausblick

Lasttests sind vor allem im Bereich der Webprogrammierung ein unverzichtbares Werkzeug zur Aufdeckung von Schwachstellen. Die in der Abteilung „DV in der Verwaltung" des CMS eingerichtete Infrastruktur auf Basis von JMeter ermöglicht eine einfache Erstellung und Ausführung von Testplänen. Somit werden Systeme wie AGNES im Vorfeld intensiv unter Lastbedingungen getestet, um den Endanwendern eine störungsfreie und somit unkomplizierte Nutzung zu bieten.

Literatur / Abbildung

[1] *The Apache Jakarta Project*. http://jakarta.apache.org/jmeter/

[2] *The Grinder, a Java Load Testing Framework*. http://grinder.sourceforge.net/

[3] *The Grinder License*. http://grinder.sourceforge.net/license.html

[4] *Eviware*. http://www.soapui.org/

[5] *Eviware*. http://www.loadui.org/

[6] *Continuent. What is Bristlecone?* http://www.continuent.com/community/lab-projects/bristlecone/

[7] *OpenSTA. Open, Systems Testing Architecture*. http://opensta.org/

[8] *CORBA*. http://www.corba.org/

[9] *WebJS. Performance und Bottleneck-Analyse*. http://www.performance-test.de/performancetest-entscheidungskriterien.htm

[10] *Neotys. Neoload – Cloud Load Testing Service*. http://www.neotys.de/services/cloud-testing-service.html

[11] *Neotys. Neoload – Performancetool und Stresstest für Webanwendungen*. http://www.neotys.de/

[12] *Compuware-Lösungen*. http://de.compuware.com/

[13] *ab – Apache HTTP server benchmarking tool*. http://httpd.apache.org/docs/2.0/programs/ab.html

[14] *Selenium*. http://seleniumhq.org/

[15] *BrowserMob*. http://browsermob.com/

[16] *Verteilte Lasttests*. Apache Software Foundation, http://jakarta.apache.org/jmeter/usermanual/jmeter_distributed_testing_step_by_step.pdf