

Evaluation von Lasttest-Tools und Performance Studie

Christian Stoll, Tim Pommerening
(Team 5)

11. März 2004

Inhaltsverzeichnis

I	Evaluation von Open-Source Lasttest-Tools	3
1	Evaluationsgrundlagen	4
2	Erläuterungen zur Evaluation	6
2.1	Apache JMeter	6
2.1.1	Voraussetzungen	7
2.1.2	Technik	7
2.1.3	Bedienbarkeit	7
2.1.4	Funktion	8
2.1.5	Sonstiges	9
2.1.6	Fazit	11
2.2	TestMaker	12
2.2.1	Voraussetzungen	12
2.2.2	Technik	12
2.2.3	Bedienbarkeit	13
2.2.4	Funktion	13
2.2.5	Sonstiges	14
2.2.6	Fazit	14
2.3	OpenSTA	14
2.3.1	Voraussetzung	14
2.3.2	Technik	14
2.3.3	Bedienbarkeit	16
2.3.4	Funktion	17
2.3.5	Sonstiges	17
2.3.6	Fazit	18
2.4	DieselTest	18
2.4.1	Voraussetzung	18
2.4.2	Technik	18
2.4.3	Bedienbarkeit	20
2.4.4	Funktion	20
2.4.5	Sonstiges	21
2.4.6	Fazit	21
2.5	Siege	21
2.5.1	Voraussetzungen	21
2.5.2	Technik	21
2.5.3	Bedienbarkeit	22
2.5.4	Funktion	22
2.5.5	Sonstiges	23
2.5.6	Fazit	23

<i>INHALTSVERZEICHNIS</i>	2
3 Evaluationsergebnis	25
3.1 Vergleich	25
3.2 Fazit	29
II Performance-Studie einer Website	30
4 Aufbau der Testumgebung	31
4.1 Verwendete Hardware	31
5 Testszenario	32
6 Testdurchführung	34
6.1 Mercury Loadrunner	34
6.2 Apache JMeter	38
6.3 Testmaker	39
6.4 OpenSTA	39
6.5 DieselTest	40
6.6 Siege	40
7 Testergebnis und Verbesserungsvorschlag	42
A Evaluation der Testwerkzeuge	44
A.1 Evaluationsfragen	44
A.1.1 Technik	44
A.1.2 Bedienbarkeit	46
A.1.3 Funktionalität	47
A.1.4 Sonstiges	50
A.2 Evaluierungsergebnisse	51
B Performance-Studie	56
B.1 Aufbau der Testskripte	56

Teil I

**Evaluation von Open-Source
Lasttest-Tools**

Kapitel 1

Evaluationsgrundlagen

Dieser Abschnitt beschreibt den Aufbau, die Durchführung und die Ergebnisse einer Evaluation von fünf Open Source Lasttest-Werkzeugen im Vergleich zum Referenzwerkzeug Loadrunner von Mercury in der Version 6.5. Als Grundlage für die Evaluation wurden die in der Diplomarbeit von Andreas Hoffmann aufgestellten Testkriterien verwendet. Da die Tests jedoch nicht nur von einer Person durchgeführt wurden, wurde nicht die der Diplomarbeit beiliegende Excel-Tabelle, sondern ein Webinterface (siehe Abbildung 1.1 auf der nächsten Seite) verwendet, um die Ergebnisse einzutragen. Das gleiche Tool erzeugte daraufhin die Ergebnisse der Evaluation (siehe Abbildung 1.2 auf der nächsten Seite). Zur Bewertung der insgesamt über 150 Testkriterien wurden die gleichen Gewichtungen verwendet wie in der Excel-Vorlage. Diese Vorlage liegt der Ausarbeitung als Arbeitsgrundlage bei. In Kapitel 2 sind die Ergebnisse der Evaluation (siehe 51) erläutert. Kapitel 3 enthält einen Vergleich der unterschiedlichen Testergebnisse und ein daraus folgender Endvergleich. Die fünf evaluierten Open-Source-Tools sind:

- Apache JMeter
- Testmaker
- OpenSTA
- Dieseltest
- Siege

Wurde bei der Evaluierung „Ja“ gewählt, so war die gefragte Funktionalität in vollem Umfang vorhanden und wurde mit der Punktzahl, die der Gewichtung entsprach gewertet. Die Auswahl von „Eingeschränkt“ bedeutet, dass die Funktionalität nicht vollständig oder nur sehr umständlich gewährleistet wird. In diesem Falle wird die Hälfte der Punktzahl einer „Ja“ Antwort gezählt. Ein Klick auf „Nein“ bedeutet, dass das Funktionsmerkmal überhaupt nicht vorhanden ist. In diesem Falle gibt es 0 Punkte. Wird „Irrelevant“ ausgewählt, so werden auch keine Punkte gezählt, es wird jedoch vermerkt, dass das Kriterium für das entsprechende Produkt unbedeutend ist. Die Möglichkeit „irrelevant“ ist z.B. dafür vorgesehen, wenn sich Evaluierungsfragen auf eine vorherige Frage beziehen und diese bereits mit „Nein“ beantwortet wurde.

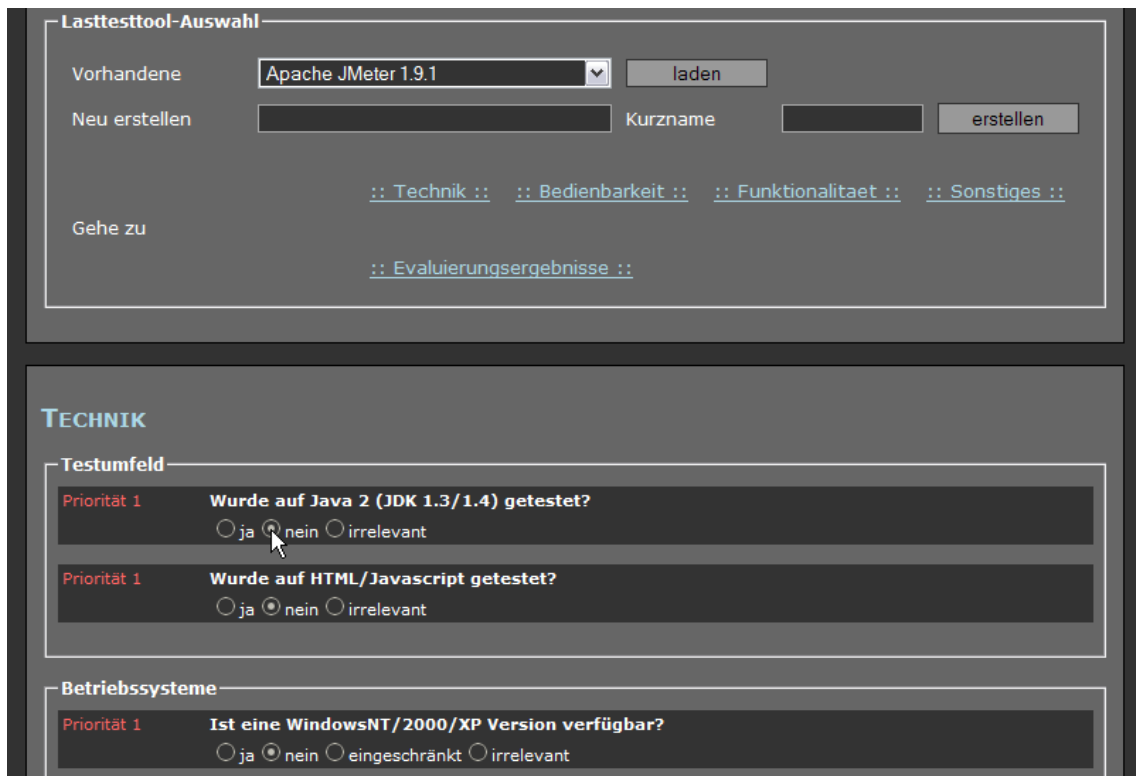


Abbildung 1.1: Webbasiertes Evaluierungstool

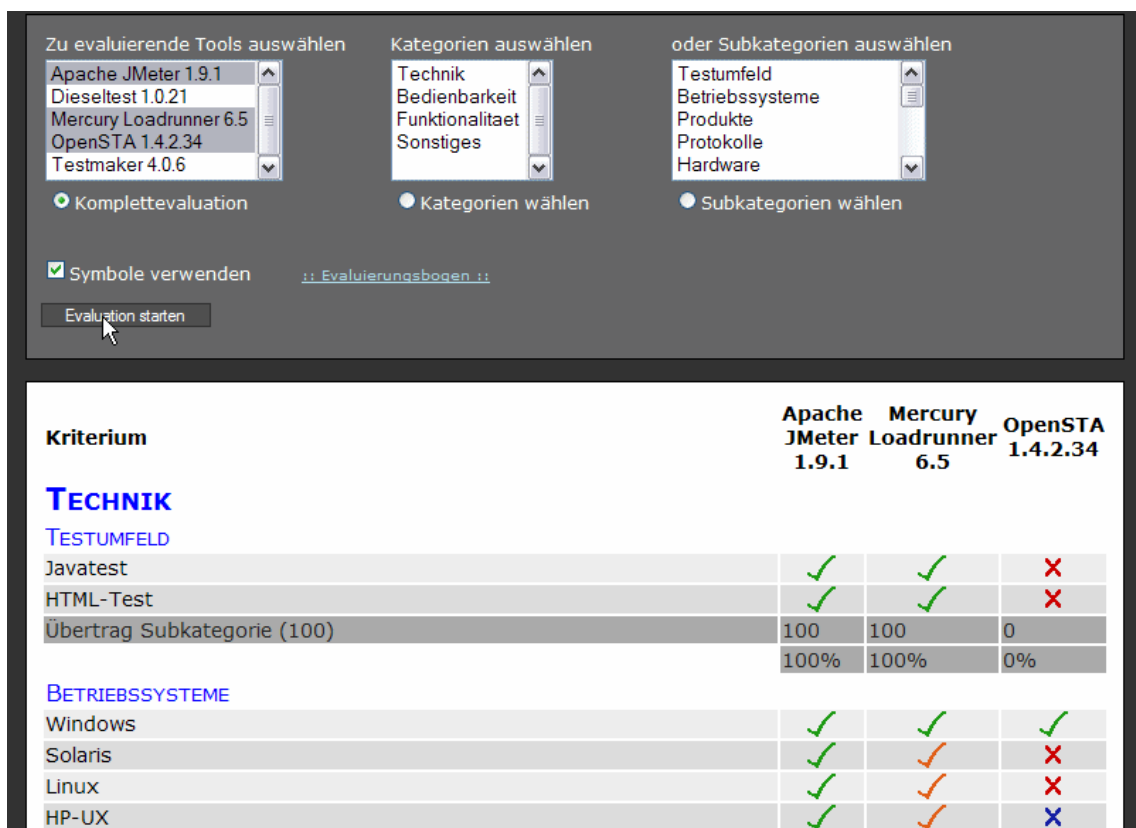


Abbildung 1.2: Webbasierte Evaluationsauswertung

Kapitel 2

Erläuterungen zur Evaluation

Einige der zu testenden Lasttestwerkzeuge sind laut deren Beschreibungen auch dazu einsetzbar, Anwendungen mittels Skripten beispielsweise auf korrekte Rückgabewerte zu prüfen. Die Tests in dieser Evaluierung beschränken sich jedoch sehr stark auf die Möglichkeit der Tests im Web. Das heißt vor allem auf das HTTP-Protokoll.

2.1 Apache JMeter

Die JMeter-Software von Apache ist eine Java Entwicklung. Das Programm wurde entwickelt, um Client/Server-Software zu testen. Laut Dokumentation¹ unterstützt JMeter sowohl das Testen von statischen als auch dynamischen Ressourcen. Angegeben wird die Testbarkeit von

- statische Dateien
- Java Servlets
- CGI Skripten
- Java Objekte
- Datenbanken
- FTP-Server
- ...

Außerdem verspricht die Dokumentation, dass JMeter ein hohes Maß an Last auf Servern erzeugen kann, um deren Stärken und ihre allgemeine Performanz zu testen. Schließlich soll es auch möglich sein, Anwendungen daraufhin zu testen, ob sie korrekte Rückgabewerte liefern. Dies soll mit Hilfe von Testskripten geschehen, in denen bestimmte Zusicherungen über die Rückgabewerte gemacht werden. Evaluiert wurde das Programm in der Version 1.9.1 für Windows. Auf der Jakarta Downloadseite² befindet sich aber auch eine Unix (Linux, Solaris, ...)-Version, wobei sich die Versionen dank Java ohnehin nicht großartig unterscheiden sollten.

¹JMeter Dokumentation jakarta.apache.org/jmeter/usermanual/intro.html

²jakarta.apache.org/site/binindex.cgi

2.1.1 Voraussetzungen

Möchte man die Testversion (1.9.1) selbst kompilieren, benötigt man mindestens das JDK 1.4, für die Ausführung wird dazu äquivalent das JRE 1.4 angegeben. JMeter verwendet als XML Parser standardmäßig den Xerces XML Parser³. Es können jedoch auch alternative Parser verwendet werden. Dazu muss dieser in den classpath hinzugefügt werden. Email-Unterstützung wird von Haus aus in kleinem Rahmen gewährleistet, falls man die JavaMail Pakete von Sun in den classpath hinzufügt. Um Webseiten zu testen, die eine SSL Verschlüsselung verwenden, muss ebenfalls eine Anpassung vorgenommen werden. Man benötigt eine Java-SSL-Implementierung. Da die zu testende Version allerdings das JRE 1.4 benötigt, kann die darin integrierte JSSE (Java Secure Socket Extension) genutzt werden. Abschließend benötigt man auch für einen JDBC Test den entsprechenden JDBC Treiber im classpath.

2.1.2 Technik

Das Werkzeug kann sowohl zum Testen von Java-, als auch von reinen HTML-Seiten verwendet werden. Da es sich selbst um ein Java Programm handelt, ist der JMeter auf allen Systemen lauffähig, für die eine Java Virtual Machine implementiert wurde. Intensiv getestet wurde das Werkzeug in einer Windows-Umgebung, die Ausführung auf einem Linux-System erwies sich jedoch als problemlos. Da laut Herstelleraussagen die Software unter Linux, Solaris und anderen Unix-Systemen getestet worden ist, wurden intensivere Tests auf diesen Plattformen aus Zeitgründen nicht mehr durchgeführt. Laut Recherchen im Internet existieren auch für die Betriebssysteme HP-UX, IBM S/390 und Tandem entsprechende JVM Implementierungen, so dass auch hier davon ausgegangen wird, dass der Apache JMeter dort lauffähig ist.

Direkte Testunterstützung für spezielle Produkte bietet der JMeter im Vergleich zum Loadrunner überhaupt nicht. Auch bei der Unterstützung der zu testenden Protokolle hängt das Referenzsystem den JMeter ab. Dieser erfüllt hierbei mit Unterstützung von HTTP, HTTPS, LDAP und JDBC nur 30% der geforderten Protokolle.

Was die benötigte Hardware angeht, so schweigt sich der Hersteller aus. Es werden ebenso keine Speicheranforderungen für virtuelle Benutzer angegeben. Beim Testen der Software stellte sich, wie es für Java-Anwendungen oft typisch ist, heraus, dass der Speicherverbrauch pro virtuellem Benutzer sehr unterschiedlich ist. Oft liegt er unter einem MB, aber gelegentlich auch darüber. Meistens jedoch wird der Speicher nach dem Beenden des Testskriptes nicht wieder freigegeben, da der Garbage Collector erst arbeitet, wenn es ihm beliebt. Das Kriterium von unter einem Megabyte pro virtuellem Benutzer konnte daher nicht als erfüllt angesehen werden. Networkmonitoring bietet der JMeter nicht an. Skriptrecording wird über einen Recording-Proxy aktiviert. Dazu muss im JMeter dem Workbench ein HTTP Proxy Server hinzugefügt werden und im Browser die Proxy-Einstellung aktiviert sein. Da standardmäßig vom JMeter alles aufgezeichnet wird, kann man noch Filter-Regeln setzen um seine Testskripte nicht etwa mit sämtlichen Bildern oder ähnlichem unnötig aufzufüllen.

2.1.3 Bedienbarkeit

Was die Installation angeht, so gestaltet sich diese für den Apache JMeter sehr einfach. Zwar gibt es weder ein Setup für Windows, noch ein Shellscript für Unix-Systeme, aber das ist im Grunde auch nicht nötig. Man entpackt das Programm in ein Verzeichnis und führt

³Apache Website <http://xml.apache.org/>

in Windows die im bin-Ordner liegende `jmeter.bat` Datei oder unter Unix das JMeter-Skript aus. Verwendet man das JDK 1.4 von Sun, so benötigt man auch keine zusätzlichen Patches oder jar-Pakete. Da es weder Einträge in die Windows Registry noch Dateien in Systemverzeichnissen gibt, gestaltet sich die Deinstallation so, dass einfach der JMeter Ordner gelöscht wird. Der Administrationsaufwand während des Betriebs ist gering bis gar nicht vorhanden. Es fiel in der verwendeten Version lediglich auf, dass ein Umstellen der Sprache von „Deutsch“, auf „Englisch“ nicht möglich war. Ein Blick in die Datei `jmeter.properties` löste das Problem, welches sich nicht als Bug sondern als Feature entpuppte. Standardmäßig ist die Sprachoption auskommentiert und es wird die Sprache der JVM verwendet. Es empfiehlt sich jedoch die Sprache auf „Englisch“ umzustellen, da die Dokumentation in Englisch ist und man bei der Beschreibung von GUI-Elementen im Deutschen nicht gleich deren Bedeutung versteht.

Die Bedienung des Tools (siehe Abbildung 2.1 auf Seite 10) geschieht im allgemeinen intuitiv. Nach kurzer Einarbeitungsphase kann man bereits einfache Skripte erstellen. Der eingeschränkte Auswahlumfang in den Menüs erleichtert die Benutzbarkeit zusätzlich. Wie auch der Loadrunner von Mercury enthält der JMeter keinen integrierten Workflow. Durch die gute Dokumentation⁴, welche allerdings nur in englisch verfügbar ist, lassen sich schnell Probleme lösen und Fragen klären. Ein Tutorium hilft in der Einarbeitungsphase. Eine Onlinehilfe ist ebenfalls integriert. Diese ist übersichtlich aufgebaut und dient als Referenz. Programmabstürze kamen bei den Tests nicht vor. Nach großen Lasttests wird allerdings sehr viel Speicher verbraucht, so dass die Benutzbarkeit durch die langsamen Reaktionen stark eingeschränkt wird.

2.1.4 Funktion

Das Aufzeichnen von Websites funktioniert, wie bereits erläutert, über einen Proxy Server. Die Aufzeichnungen selbst sind recht unkompliziert. Allerdings ist es etwas kompliziert, Teile einer Aufzeichnung zu gruppieren. Nach dem Aufzeichnen zeigt JMeter das Testskript als Baumstruktur an. Möchte man Parameter von Hand ändern, so klickt man auf den entsprechenden Eintrag im Baum und erhält im rechten Fenster alle Infos über die Seite inklusive übergebener Parameter. Als Skriptsprache dient das zur Zeit überall favorisierte XML (Beispielskript, siehe auf Seite 10). Man kann die Skripte also mit XML Kenntnissen auch manuell bearbeiten, was sich allerdings nicht unbedingt anbietet, da man über die JMeter-Bearbeitung die Einstellungen recht komfortabel ändern kann. Durch die XML-Verschachtelungen werden Blöcke gruppiert. Innerhalb des Tools wird dies durch Controller-Elemente realisiert. Es gibt verschiedene Controller, Standard-Controller z.B. dienen nur als Gruppierung, darüber hinaus gibt es auch Aufnahme-Controller, die die Proxy-Daten aufzeichnen oder Zufalls-Controller, die für jeden Durchlauf einen zufälligen Eintrag aus ihrer Liste auswählen. XML-Kommentare kann man natürlich auch in die Skripte einfügen, jedoch nicht während der Aufzeichnung. Eine Möglichkeit, Synchronisationspunkte festzulegen, wurde nicht gefunden. Die Möglichkeit, mehrere Testskripte in einer Umgebung zu öffnen und auszuführen besteht, allerdings kann man in einer Recording-Session nicht mehrere erzeugen. Dazu muss man die Session beenden. Man kann gespeicherte Skripte einfach in eine Testumgebung einladen. Jedes Skript bildet eine so genannte Thread-Gruppe, die unabhängig von, jedoch gleichzeitig mit anderen Skripten ausgeführt werden kann. Für jede Thread-Gruppe kann festgelegt werden, wie viele Benutzer simuliert werden. Außerdem kann man einstellen, wie oft die Skripte wiederholt werden sollen und ob und wie ein Ramp-Up durchgeführt wird.

⁴JMeter Dokumentation <http://jakarta.apache.org/jmeter/usermanual/index.html>

Nach dem Aufzeichnen kann man die Skripte natürlich abspeichern. Man sollte darauf achten, dass man immer die Dateierweiterung an den Skriptnamen anhängt, da sonst das Skript ohne Erweiterung gespeichert wird und später nur schwer wieder gefunden werden kann. Aus diesem Grund wurde der Punkt „Übersichtliches Ablegen der Skripte“ auch nur mit „eingeschränkt“ bewertet.

Eine Parametrisierung der Eingabewerte ist möglich. Dabei können verschiedene Funktionen, wie z.B. die `__random()` Funktion, verwendet werden, die eine Zufallszahl innerhalb eines gewünschten Bereichs zurück gibt. Die Verwendung von Datentabellen ist ebenso möglich. Diese können allerdings lediglich mit einem externen Texteditor erzeugt werden. Die manuelle oder automatische Erweiterung bereits aufgezeichneter Skripte ist möglich. Das Trennen oder Zusammenführen von Skripten ist nur manuell möglich. Es existiert kein entsprechendes Werkzeug dafür. Insgesamt erhält das Tool in der Unterkategorie „Testskripterstellung“ eine Bewertung von 67,9%.

Der Apache JMeter unterstützt keine Art des Testmanagements. Ein Testszenario wird zentral über einen Klienten gesteuert. Ein JMeter Klient bietet die Möglichkeit über die „Remote Start“ und „Remote Stop“ Funktion mehrere Klienten zu steuern. Mehrere Benutzergruppen in einem Szenario sind ebenso möglich. Diese werden hier, wie bereits erwähnt, Thread-Gruppen genannt. Bestimmte Bandbreiten können während des Testens nicht simuliert werden. Es existiert über den Proxy jedoch die Möglichkeit, ins Internet zu testen. Da kein Absturz beim Testen erfolgte, bekommt JMeter genau wie das Referenzmodell im Punkt „Testfortführung nach Absturz“ volle Punkte. Die Messdaten sind während des Testens beobachtbar und können, sofern im Graph vorhanden, ein- und ausgeblendet werden. Ablegen der Messdaten erfolgt wiederum in einer XML-Datei. Das Ablegen der Daten in CSV oder über ODBC ist nicht möglich, da die neueren Excel-Versionen XML-Formate lesen können, ist dies auch nicht zwingend notwendig. Testläufe des Skripts sind insofern möglich, als dass man die Anzahl der Benutzer und der Durchläufe auf 1 herabsetzt. Ein unterstützendes Scriptdebugging enthält das Tool nicht. Im Unterbereich „Testablauf“ erhält Apache JMeter 56,3 Prozentpunkte.

Die Testauswertung fällt beim Apache JMeter äußerst dürftig aus. Zwar kann man alle Testergebnisse und sogar Graphen im XML-Format abspeichern, jedoch ist die Anzeigefreundlichkeit und das automatische Auswerten der Daten mangelhaft. In einem Graphen kann man sich „Daten“, „Durchschnitt“, „Median“, „Abweichung“ und „Durchsatz“ anzeigen lassen (siehe Abbildung 2.2). Man kann es aber auch genauso gut bleiben lassen, denn man bekommt beim Ansehen der grafischen Auswertung schnell das Gefühl, dass die Hersteller einfach nur ein wenig Grafik in den tristen Lasttest-Alltag bringen wollten. Der Graph ist weder skalierbar, noch werden Informationen darüber angezeigt, was auf der x-Achse überhaupt abgetragen wird. Hier wäre es z.B. sinnvoll anzuzeigen, wie viele Benutzer gerade aktiv sind und wann ein Benutzer fertig ist. Zu den eben genannten Messdaten kommen dann noch gelbe Punkte auf dem Graphen hinzu, die nirgends erklärt sind. Vergleiche von unterschiedlichen Testreihen können nicht durchgeführt werden. Immerhin können die Testergebnisse in Excel als XML-Format geladen und dort dann wirklich ausgewertet werden. In der Unterkategorie Testauswertung erhält JMeter daher auch nur 33,9%. Messdaten über die Maschine und das Betriebssystem kann JMeter übrigens in keiner Form anzeigen.

Insgesamt liegt Apache JMeter in der Kategorie „Funktionalität“ bei 51%.

2.1.5 Sonstiges

Apache bietet als Support lediglich Mailinglisten an, was für ein Open-Source-Produkt nichts ungewöhnliches ist. Da die Entwickler aktiv an den Mail-Anfragen teilnehmen ist

Algorithm 1 Apache JMeter Testskriptausschnitt: Timer-Element

```

...
<node>
  <testelement class="org.apache.jmeter.timers.GaussianRandomTimer">
    <property propType="org.apache.jmeter.testelement.property.StringProperty"
      name="TestElement.name" xml:space="preserve">Gaussian Random Timer</property>
    <property propType="org.apache.jmeter.testelement.property.StringProperty"
      name="TestElement.test_class" xml:space="preserve">
      org.apache.jmeter.timers.GaussianRandomTimer</property>
    <property propType="org.apache.jmeter.testelement.property.BooleanProperty"
      name="TestElement.enabled" xml:space="preserve">true</property>
    <property propType="org.apache.jmeter.testelement.property.StringProperty"
      name="ConstantTimer.delay" xml:space="preserve">3000</property>
    <property propType="org.apache.jmeter.testelement.property.StringProperty"
      name="RandomTimer.range" xml:space="preserve">2000</property>
    <property propType="org.apache.jmeter.testelement.property.StringProperty"
      name="TestElement.gui_class" xml:space="preserve">
      org.apache.jmeter.timers.gui.GaussianRandomTimerGui</property>
  </testelement>
</node>
...

```

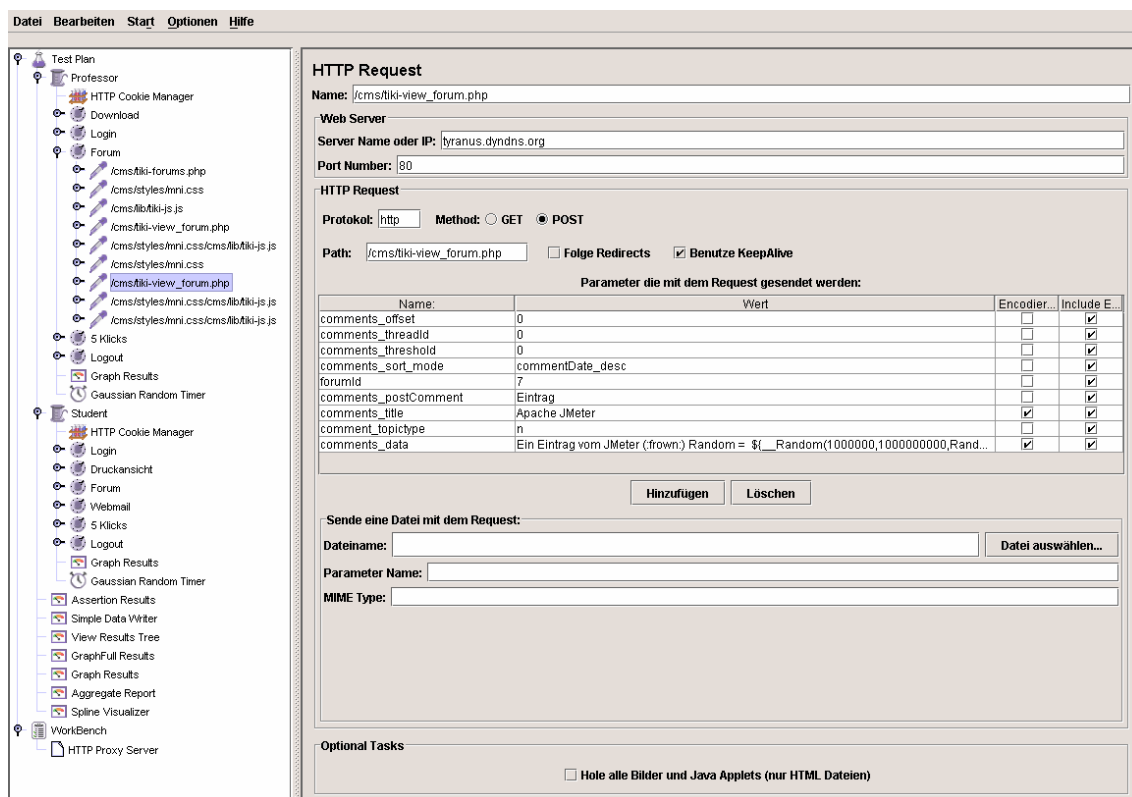


Abbildung 2.1: Apache JMeter - Übersicht

anzunehmen, dass der Support entsprechend schnell und fachlich untermauert ist. Über die Einsatzbreite des Produkts konnte keine Aussage getroffen werden.

In der Kategorie „Sonstiges“ erhält der JMeter 26,8%.

2.1.6 Fazit

Durch die Verwendung von Java eignet sich Apache JMeter für beinahe jede Plattform. Umständliches Installieren und Konfigurieren entfällt. Wegen der kurzen Einarbeitungszeit, die die gute Dokumentation gewährleistet, kann man recht schnell kleine Testskripte erstellen. Die Protokollunterstützung richtet sich vor allem auf Webanwendungen. In der Bedienbarkeit kann JMeter sogar das Referenztool Loadrunner schlagen. Das Werkzeug eignet sich sowohl zum testen statischer, als auch dynamischer Seiten. HTTP-Parameter sind parametrisierbar und bestimmte Funktionen können auch in den Testskripten verwendet werden. Der JMeter konnte das in Teil auf Seite 30 geforderte Testszenario vollständig erfüllen. Mangelhaft ist jedoch die Auswertung der Tests. Die angezeigten Graphen sind wenig aussagekräftig und können kaum bis gar nicht an die Bedürfnisse des Benutzers angepasst werden. Vergleiche zwischen unterschiedlichen Lasttests sind nicht möglich. Das einzig Positive an der Testauswertung ist, dass sämtliche Daten im XML-Format gespeichert werden, welche bspw. in Excel geladen und dort grafisch ausgewertet werden können. Zu diesem Problem kommen noch die vielen nicht unterstützten Protokolle und Produkte, bei denen JMeter an Boden verliert. Die Abschließende Wertung von Apache JMeter liegt mit 2295 von 4660 Punkten bei 49,2%

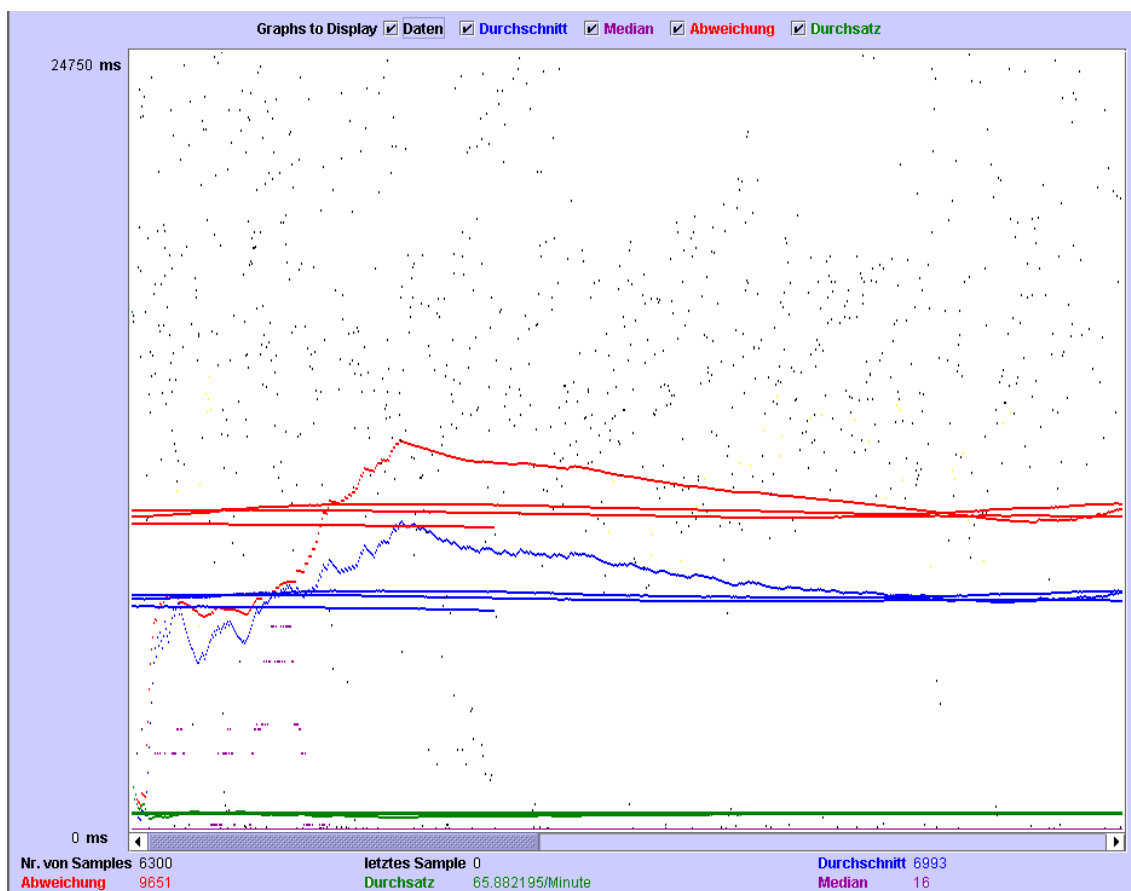


Abbildung 2.2: Apache JMeter - Graphische Auswertung

2.2 TestMaker

TestMaker lag zum Test in der Version 4.0.6 vor. Dieses Programm nimmt im Testfeld eine Ausnahmestellung ein, da es uns mit diesem Produkt nicht gelang, einen Lasttest durchzuführen. Die Homepage⁵ hinterlässt einen zwiespältigen Eindruck: Auf den ersten Blick erscheint sie gut gegliedert und informativ, beim genaueren Hinsehen jedoch fällt auf, dass die Firma *PushToTest Global Services* offensichtlich mit dem TestMaker Geld verdienen möchte. So bietet sie allerlei Dienstleistungen rund um ihr Open-Source-Produkt an, und die Hilfe auf der Homepage ist derart spärlich, als wollte man möglichst nicht zu viel Wissen zu dem Produkt preisgeben. Es wird nur auf verschiedene Bücher zum Testmaker verwiesen, außerdem wird ein weiteres kostenpflichtiges Produkt namens „TestNetwork“ angepriesen. Es gibt ein kleines Mini-Tutorium für den TestMaker, in dem gezeigt wird, wie mit TestMaker ein Skript erstellt werden kann. Dies ist jedoch so einfach gehalten, dass es spezielle Probleme wie Cookies oder die Parametrisierung von Daten komplett außen vor lässt. Laut der Homepage soll es auch möglich sein, mit TestMaker einen Lasttest durchzuführen, über das *Wie* erfährt man allerdings nichts. Es scheint so, als würde der TestMaker erst in Verbindung mit dem kostenpflichtigen TestNetwork ein vollwertiges Lasttesttool ergeben.

2.2.1 Voraussetzungen

TestMaker ist eine Java Entwicklung, es benötigt zum Ausführen die Java-Runtime in der Version 1.4 oder höher. Damit ist eine hohe Portabilität gegeben. Über weitere Voraussetzungen schweigt sich die Homepage aus, es wird keinerlei Mindestanforderung an Speicherkapazität oder Rechenleistung gegeben. Man vertraut wohl darauf, dass da, wo die JVM in der Version 1.4 lauffähig ist, genügend Rechenpower für den TestMaker vorhanden ist. Immerhin ist die Installation denkbar einfach: Programm runterladen, entpacken und mit `Testmaker.bat` wird das Programm gestartet.

2.2.2 Technik

Mit dem Programmstart von TestMaker wird auch ein eigener Proxy gestartet, den man im gewünschten Browser einstellen muss. Damit ist TestMaker browserunabhängig. Der Proxy läuft standardmäßig auf dem Port 8090, im Test gab es mit dieser Konfiguration keinerlei Probleme, sowohl der *Internet Explorer* als auch *Mozilla* in der Version 1.6 konnten nach Einstellen dieses Proxys zum Aufzeichnen genutzt werden. Das Aufzeichnen von Skripten ist denkbar einfach, man erstellt mit einen neuen „HTML Agent Recorder“ und sofort beginnt TestMaker, alles, was über den eigenen Proxy aufgerufen wird, mitzuschneiden. Als Skriptsprache kommt *Jython*⁶, die auf Java basierende Variante der bekannten Skriptsprache *Python*⁷ zum Einsatz. Hier hat man also eine mächtige und ausgereifte Skriptsprache zur Hand, die vielfältigste Möglichkeiten der Kommentierung und Strukturierung bietet. Nach dem Aufzeichnen kann das Skript gespeichert werden, allerdings muss man hier beachten, der Datei die Endung `*.a` zu geben, da dies nicht automatisch geschieht. Vergisst man dies, sucht man sein eben gerade aufgezeichnetes Skript zunächst vergeblich und TestMaker kann mit dieser Datei auch nichts anfangen. Zur Kontrolle kann das Skript noch einmal abgespielt werden, aber hier ist Vorsicht geboten. Der TestMaker spielt das Script sofort und ohne Nachfrage 50 mal hintereinander ab - wir haben bis

⁵www.pushttotest.com/ptt

⁶www.jython.org

⁷www.python.org

zum Ende der Evaluation nach einer Möglichkeit gesucht, dieses Verhalten zu beeinflussen, leider jedoch ohne Erfolg. Überhaupt scheint der TestMaker hiermit seine Funktionalität erfüllt zu haben, man findet keinerlei Einstellmöglichkeiten für die Planung eines Lasttests, auch die Homepage schweigt sich darüber aus. Auch das erfolgreiche Beenden eines Skriptes quittiert der Testmaker lediglich mit ein paar minimalistischen Daten in Textform und einem „Done“.

2.2.3 Bedienbarkeit

Testmaker lässt sich relativ gut und intuitiv bedienen - allerdings bietet es auch außer den Möglichkeiten zur Skriptaufzeichnung keinerlei weitere Bedienungsmöglichkeiten. Die offensichtlich fehlende Möglichkeit, ein Skript zur Kontrolle ein einziges mal abzuspielen und während des Testlaufs eventuell Debugging-Ausgaben zur erhalten, ist als absolut mangelhaft zu bezeichnen. Ebenso mangelhaft ist die rund um die Funktionalität des TestMakers angebotene Hilfe - so hinterlässt TestMaker den Eindruck, als könne es wesentlich mehr, jedoch fehlt dazu die geeignete Dokumentation. Hier ist klar zu erkennen, dass die Firma PushToTest Global Services offensichtlich mit teuren Schulungen und ihrem kostenpflichtigen Zusatzprodukt TestNetwork Geld verdienen will.

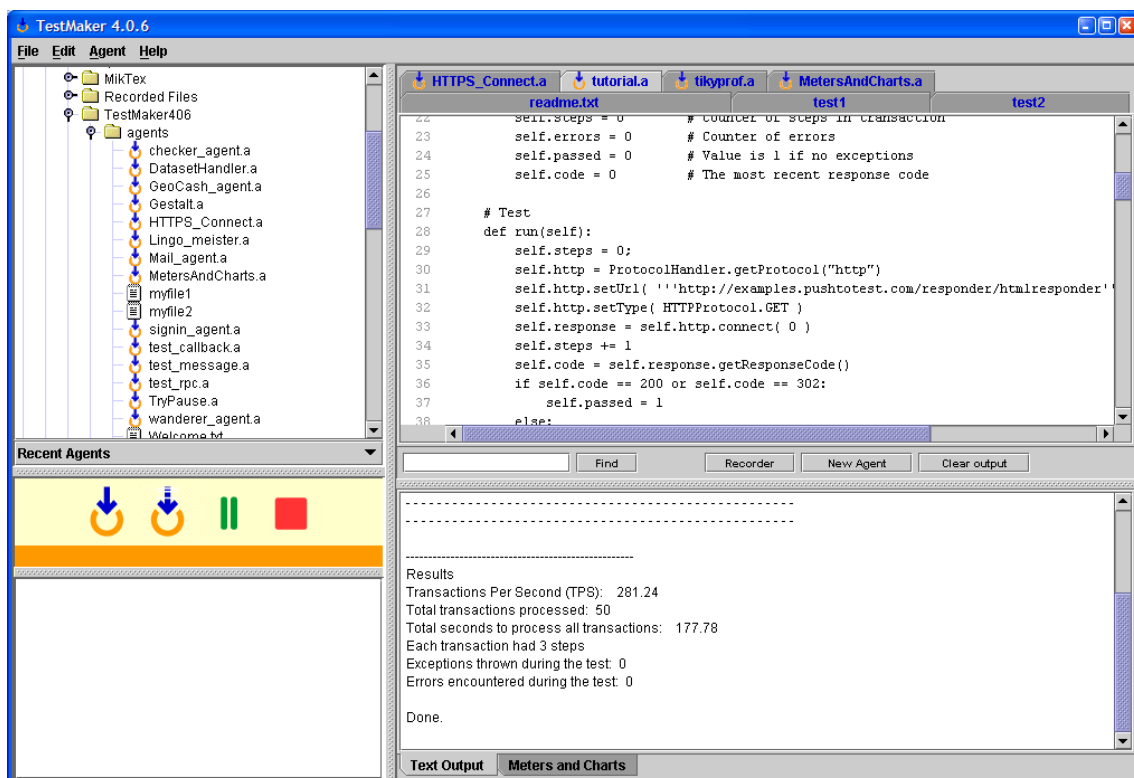


Abbildung 2.3: Testmaker nach erfolgreichem Abspielen eines aufgezeichneten Skriptes

2.2.4 Funktion

Bei der Funktionalität leistet sich TestMaker einen herben Patzer: Das von uns erstellte Testszenario ließ sich zwar aufzeichnen, beim Abspielen allerdings trat immer eine „Cookie Malform Exception“ auf. Leider war es uns bis Testende aufgrund der spärlich angebotenen Dokumentation nicht möglich, diesen Fehler zu beheben. Offensichtlich bereiteten dem

TestMaker die vom Portal gesetzten Cookies Probleme, eine genauere Aussage darüber ist leider nicht möglich. Von der fehlenden Möglichkeit, einen Lasttest mit sinnvollen Testergebnissen durchzuführen, wurde ja bereits in der Einleitung zu Testmaker berichtet.

2.2.5 Sonstiges

Der angebotene Support ist nicht gerade üppig, immerhin gibt es ein „User-Forum“ und ein „Developer-Forum“, allerdings wird einem hier zunächst eine gültige EMail-Adresse abverlangt. Weiterhin gibt es eine kostenpflichtige Support-Telefonnummer. Da auch die gebotene Onlinehilfe und das Tutorium uns bei unserem Test nicht wirklich weiterhelfen konnte, wurde TestMaker in dieser Kategorie eher schlecht bewertet.

2.2.6 Fazit

TestMaker scheint durchaus Potential als Lasttest-Tool zu besitzen, vor allem der Einsatz der Skriptsprache *Jython* scheint ein interessantes Feature zu sein. Leider blieben uns die Möglichkeiten des TestMakers aufgrund der unzureichenden Dokumentation jedoch im Testbetrieb verwehrt. Man muss hier offensichtlich bereit sein, das kostenpflichtige Zusatzprodukt TestNetwork und gleich eine ganze Reihe von Schulungen dieser beiden Produkte zu erwerben - das dies auch anders geht, beweist der nachfolgende Testkandidat OpenSTA.

2.3 OpenSTA

2.3.1 Voraussetzung

Das Produkt OpenSTA (Open Sytems Test Architecture) ist nach der GNU GPL lizenziert. Es eignet sich dazu, HTTP/HTTPS-Last zu erzeugen. Als Plattform steht nur Win32 zur Verfügung, eine Linux-Version gibt es nicht. Laut Angaben der Homepage wurde es mit Microsoft Visual Studio 6 entwickelt.

2.3.2 Technik

OpenSTA teilt sich ganz ähnlich wie der Mercury Loadrunner strikt in zwei Teile auf: Zunächst werden Skripte erzeugt, und mit diesen Skripten können dann Tests geplant und durchgeführt werden. Zum Erzeugen der Skripte wird aus dem Hauptprogramm OpenSTA „Commander“ ein Unterprogramm namens „Script Modeler“ gestartet, der ausschließlich dazu dient, Skripte aufzuzeichnen und zu bearbeiten. Im Script Modeler wird dazu zunächst der gewünschte Webbrowser festgelegt, es besteht die Auswahl zwischen dem *Microsoft Internet Explorer* (Version 4, 5 und 6) und *Netscape*. Zum Aufzeichnen startet OpenSTA zusätzlich einen eigenen Nameserver, der sich allerdings nur durch ein Icon im System-Tray bemerkbar macht und auch keinerlei Einstellmöglichkeiten bietet. Nach dem Festlegen des Browsers kann das Aufzeichnen des Skripts beginnen. Ein Klick auf „Record“ im Script Modeler startet automatisch den eingestellten Webbrowser und zeichnet alle Änderungen in diesem Browser auf. Das Aufzeichnen des Skriptes kann pausiert und später fortgesetzt werden. Der Pause-Modus hält die Aufzeichnung lediglich an - man sieht jedoch nicht, was bisher schon aufgezeichnet wurde. Ein Klick auf Stop unterbricht die Aufzeichnung und zeigt den generierten Code des Script Modelers an. Mit „Play“ kann das eben aufgezeichnete Skript zum Test noch einmal abgespielt werden. Leider ist es nicht möglich, die „Think-Times“ beim erneuten Abspielen einfach auszuschalten,

sie lassen sich nur durch direkten Eingriff in den generierten Code einzeln ändern. Beim Abspielen des Skriptes muss der Anwender daher Geduld aufbringen und seine eigenen Think-Times abwarten, falls er das Skript nicht verändern möchte. Der Code wird in einer proprietären Sprache, der Script Control Language (SCL), erzeugt. Diese ist jedoch recht einfach und schnell zu verstehen und wird zusätzlich durch die Syntaxhervorhebung des Script Modelers vereinfacht. Der Script Modeler erlaubt zusätzlich das Erzeugen von Variablen, um beispielsweise beim Test einer Seite mit Login bei jedem Testdurchlauf verschiedene Login-Daten zu benutzen. Die Daten können entweder ähnlich wie in einem Array direkt im Code angegeben werden, sie können jedoch auch in einer Datei gespeichert werden oder aus einer ODBC-Datenbank stammen. Hierzu hält der Script Modeler beim Erzeugen von Variablen einen Assistenten bereit, mit dem entweder automatisch neue Werte generiert werden können oder die Datenherkunft für die Variable festgelegt werden kann. Nach erfolgreichem Erzeugen und individuellen Anpassen des Skriptes kann

```

Entry[USER_AGENT,USE_PAGE_TIMERS]

Start Timer T_TIKIPROF
PRIMARY GET URI "http://tyranus.dyndns.org/cms HTTP/1.0" ON 28
HEADER DEFAULT HEADERS
,WITH {"Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*",
"Accept-Language: de",
"Cookie: "+S_cookie_28_0+"; "+S_cookie_28_1,
"Connection: Keep-Alive"}

WAIT 532

PRIMARY GET URI "http://tyranus.dyndns.org/cms/ HTTP/1.0" ON 28
HEADER DEFAULT HEADERS
,WITH {"Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*",
"Accept-Language: de",
"Cookie: "+S_cookie_28_0+"; "+S_cookie_28_1,
"Connection: Keep-Alive"}

DISCONNECT FROM 28

WAIT 1765

PRIMARY GET URI "http://tyranus.dyndns.org/cms/tiki-index.php HTTP/1.0" ON 29
HEADER DEFAULT HEADERS

```

```

!Begin replay
1-1 : [64]:START TIMER: T_TIKIPROF
1-1 : [65]:REQUEST(28): GET http://tyranus.dyndns.org/cms HTTP/1.0
1-1 : [72]:WAIT : Delay: 532
1-1 : [72]:Connection 28: receiving results with status 301
1-1 : [74]:REQUEST(28): GET http://tyranus.dyndns.org/cms/ HTTP/1.0
1-1 : [81]:DISCONNECT(28)

```

Abbildung 2.4: OpenSTA Script Modeler beim Abspielen eines aufgezeichneten Skriptes

nun ein Lasttest, der auf mindestens einem Skript besteht, geplant und durchgeführt werden. Zur Durchführung des Lasttests kann nun im OpenSTA Commander ein Test mit den erstellten Skripten vorbereitet werden. Dazu werden die Skripte, aus denen sich der Lasttest aufbauen soll, einfach per Drag and Drop aus der Baumstruktur auf das Teststellungs-fenster gezogen. Hier kann dann noch festgelegt werden, mit wie vielen VUs ein

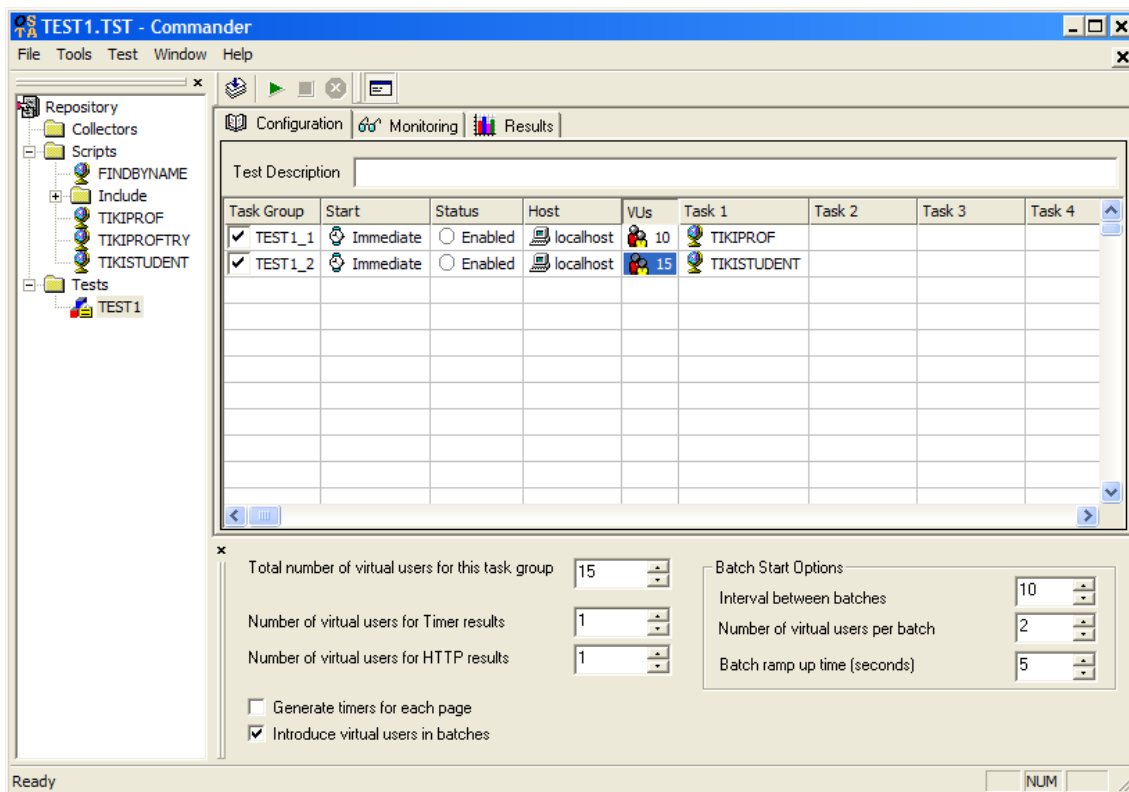


Abbildung 2.5: Lasttestvorbereitung im OpenSTA Commander

Skript abgespielt werden soll. Es ist ein Ramp-Up möglich, sowie zeitversetztes Starten der einzelnen Skripte. Während des Lasttests können zudem einzelne Skripte vom Benutzer gezielt gestoppt werden. Auch kann man die Dauer auf einen Durchlauf beschränken oder eine beliebige Testdauer festlegen. Zusätzlich ist es sogar möglich, einen verteilten Lasttest zu starten, hierzu muss dann die Einstellung „localhost“ bei den auf die IP des Rechners geändert werden, von dem das entsprechende Skript gestartet werden soll. Auf dem Zielrechner muss dann OpenSTA ebenfalls installiert und zu Testbeginn gestartet sein. Ist das Erstellen des Testszenarios abgeschlossen, kann durch Klick auf „Play“ der Test gestartet werden. Während des Tests sind allerdings nur minimale Informationen über den Testablauf verfügbar, die wichtigen grafischen Auswertungen sind erst nach Testende verfügbar. OpenSTA speichert automatisch die ermittelten Testergebnisse mit Datum und Uhrzeit, so das man später vergleichen kann. Auch ist es möglich, die Testergebnisse in eine Excel-Datei zu exportieren, jedoch muss man diese noch umrechnen, da OpenSTA offensichtlich das amerikanische Zahlenformat (mit Punkt als Trennzeichen) benutzt und die von uns eingesetzte deutsche Excel-Version damit ihre Probleme hatte. Die von OpenSTA erzeugten Graphen lassen sich frei skalieren, allerdings lässt sich pro Graph immer nur eine Messkurve darstellen, was das Vergleichen mit anderen Testergebnissen erschwert.

2.3.3 Bedienbarkeit

OpenSTA lässt sich recht intuitiv bedienen. Es bietet eindeutige Symbolleisten und ist nicht mit Einstellmöglichkeiten überladen. Die Hilfe ist in ihrem Umfang ebenfalls nicht überladen, jedoch keinesfalls zu knapp. Es gibt ein Tutorium, mit dem man die Möglichkeiten von OpenSTA gut erlernen kann. Der Script Modeler lässt dennoch an einigen Punkten

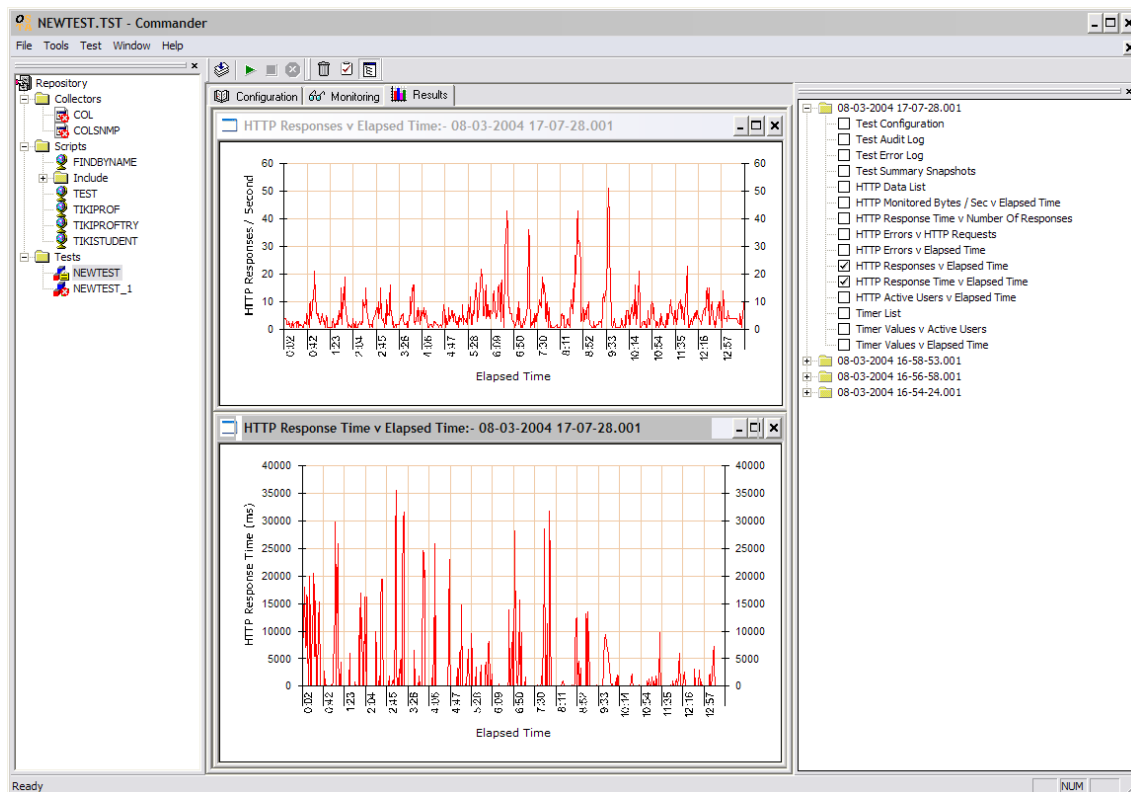


Abbildung 2.6: Ergebnisauswertung mit OpenSTA

Wünsche offen, so zum Beispiel das fehlende Abschalten oder Variieren der Think Times. Das Erstellen des eigentlichen Testszenarios hingegen ist sehr komfortabel, mit wenigen Einstellungen ist es bereits möglich, einen umfangreichen Lasttest zu planen. Bei der Testauswertung patzt OpenSTA dann leider wieder, zwar werden Testergebnisse übersichtlich gespeichert, jedoch muss man sich immer bis zur Fertigstellung des Testes gedulden, bevor man Ergebnisse einsehen kann. Graphen, die während des Tests dynamisch mit Werten gefüllt werden, sucht man vergebens. Auch der Excel-Export hat seine Tücken. Dennoch nimmt OpenSTA in Puncto Bedienbarkeit aufgrund der einfachen Skript- und Testerstellungen einen Spitzenplatz im Test ein.

2.3.4 Funktion

OpenSTA lief im Testbetrieb sehr stabil und machte einen ausgereiften Eindruck. Der Script Modeler zeichnete ohne Probleme das von uns erstellte Testszenario auf, dies gelang im Testfeld sonst nur dem kostenpflichtigen Mercury Loadrunner. Auch der Lasttest selbst lief auf Antrieb ohne Probleme. Das Programm gehörte zu den resourceschonendsten im Testfeld - gönnte es sich trotz 25 VUs während des Lasttests gerade mal 10 MB Hauptspeicher unseres Windowssystems.

2.3.5 Sonstiges

Der Support bei OpenSTA scheint in Ordnung zu sein. Es gibt eine eigene Community-Seite, in denen z.B. über OpenSTA im täglichen Einsatz berichtet wird und Benutzer fleißig ihre Erfahrungen austauschen. Dennoch kann sicher nicht ein so umfassender Support

geboten wie es beispielsweise beim Loadrunner der Fall ist - der hier gebotene Support ist dagegen aber auch kostenlos.

2.3.6 Fazit

OpenSTA gehört nach dem Loadrunner zu den komfortabelsten Produkten unseres Tests. Es bietet zweifelsohne für ein Open-Source-Produkt sowohl einen erstaunlichen Funktionsumfang als auch eine bemerkenswerte Produktreife. Zwar sind im Detail durchaus Verbesserungen wünschenswert, dies ändert jedoch nichts daran, das sich OpenSTA hervorragend für Lasttests größeren Ausmaßes eignet und bis zu einem gewissen Grad sogar, vor allem in Anbetracht der Kosten, eine Alternative zum teuren Loadrunner von Mercury darstellt. Das gegenüber JMeter schlechte Abschneiden im Test resultiert aus der Beschränkung auf die Win32-Plattform - in Puncto Leistung und Komfort ist es JMeter auf jeden Fall überlegen.

2.4 DieselTest

Gleich zu Beginn sei gesagt, das die Homepage von DieselTest mittlerweile nicht mehr erreichbar ist. DieselTest ist nach der GNU GPL lizenziert und lag zum Test in der Version 1.0.21 vor. Diese Version stammt aus dem Jahr 2001, es scheint also so, als würde dieses Produkt nicht mehr weiterentwickelt. Es ist zwar ein sehr simples, dennoch lauffähiges Lasttesttool, jedoch spiegelt sich die fehlende Homepage und die damit mangelhafte Produktdokumentation in den Testergebnissen zweifelsohne wieder.

2.4.1 Voraussetzung

DieselTest ist ähnlich wie OpenSTA ein reines Win32-Produkt. Da, wie bereits erwähnt, die DieselTest-Homepage nicht mehr erreichbar ist, kann an dieser Stelle auch nichts verifiziertes über etwaige weitere Voraussetzungen gesagt werden.

2.4.2 Technik

DieselTest verfügt über ein Benutzerinterface, welches auf ein absolutes Minimum beschränkt ist - es wirkt gerade zu puristisch. Nach dem Start des Programms kann man mit dem Button „Create New Script“ die Aufzeichnung eines Testskripts starten. DieselTest startet dazu einen eigenen Webbrowser, in dem es die Änderungen aufzeichnet. Nach Stoppen der Aufzeichnung speichert DieselTest das Skript. Das Skript selbst liegt nicht in einer Code-ähnlichen Form vor, sondern in einer Tabellenstruktur, in der jeder HTTP/HTTPS-Request aufgelistet wird. Die Editiermöglichkeiten sind beschränkt, es ist beispielsweise nicht einmal möglich, einen Kommentar einzufügen. Nach Aufzeichnung des Skripts kann der Lasttest auch sofort beginnen: Man kann die Zahl der Benutzer einstellen, die Dauer des Tests und eine Mögliche Ramp-Up-Zeit. Weiterhin steht zur Auswahl, ob die während der Aufzeichnung gespeicherte Think-Time oder eine fest einstellbare Think-Time genutzt werden soll. Ein Test kann außerdem mehrere Skripte enthalten. Mit „Run Test“ wird der Lasttest gestartet; es öffnet sich ein neues Fenster, indem ein kleiner Chart dargestellt wird sowie weitere statistische Daten angezeigt werden (Anzahl der Benutzer, Start und Dauer des Tests, Anzahl der aufgetretenen Fehler und die CPU-Auslastung des Rechners). Der Chart stellt die durchschnittliche Seitenaufbauzeit in Relation zu der Benutzeranzahl dar. Ist der Test fertig, kann der Chart als Grafik in die Zwischenablage kopiert werden - dies

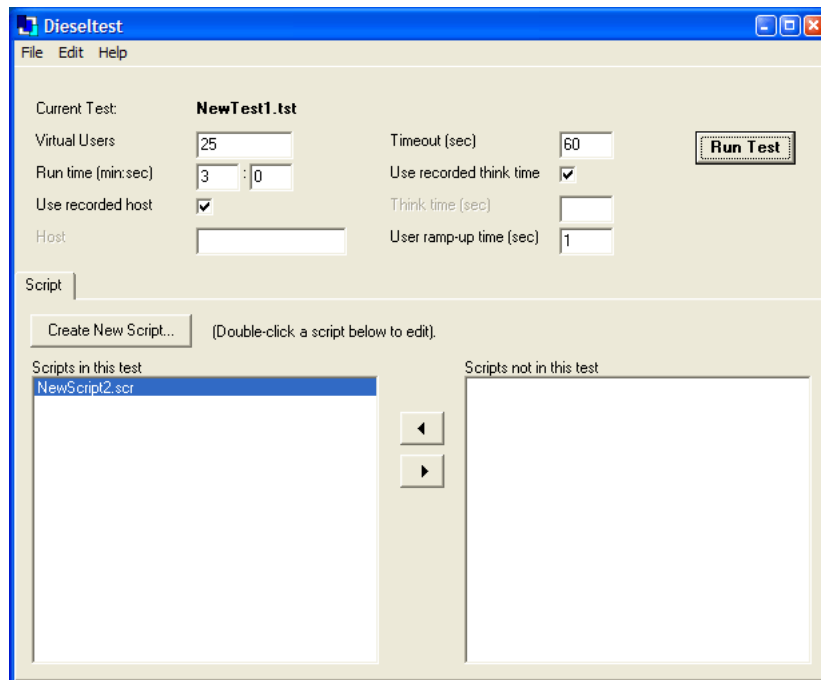


Abbildung 2.7: Die minimalistisch anmutende Benutzeroberfläche von Dieseltest

The screenshot shows the 'Edit script: NewScript2.scr' window. It has a title bar 'Edit script: NewScript2.scr' and a menu bar with 'Add', 'Delete', 'Save', and 'Close'. Below the menu bar is a table with columns: 'ThinkTime', 'Protocol', 'Host', 'PageRef', and 'Method'. The table contains 16 rows of data. The first row is selected and highlighted in blue.

ThinkTime	Protocol	Host	PageRef	Method
22375	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms	GET
547	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/	GET
2219	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/tiki-index.php	GET
2844	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/styles/mni.css	GET
875	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/lib/tiki-js.js	GET
1031	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/styles/mni/img/top_22.jpg	GET
0	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/styles/mni/img/top_22_hor.jpg	GET
640	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/styles/mni/img/top_22_vert.jpg	GET
32	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/img/icons/ico_link.gif	GET
672	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/img/css1.png	GET
31	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/img/wiki_up/axel.jpg	GET
515	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/img/icons/ico_print.gif	GET
32	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/img/valid-xml10.png	GET
703	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/img/php.png	GET
15	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/img/pear.png	GET
516	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/img/smarty.gif	GET
5719	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/tiki-login.php	POST
1875	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/tiki-index.php	GET
2515	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/styles/mni.css	GET
579	HTTP	tyranus.dyndns.org	http://tyranus.dyndns.org/cms/lib/tiki-js.js	GET

Abbildung 2.8: Skriptdarstellung ala Dieseltest

ist jedoch die einzige Art der Testauswertung, es gibt keinerlei Möglichkeit, an präzisere Daten über den Test heranzukommen.

2.4.3 Bedienbarkeit

Die Bedienbarkeit von Dieseltest gehört zu den einfachsten im Testfeld - beschränkt sich doch das gesamte Benutzerinterface auf ein notwendiges Minimum. Doch bleibt der Wunsch nach mehr Funktionalität offen, es gibt keine Möglichkeit, variable Daten in den Test einfließen zu lassen oder das aufgezeichnete Skript sinnvoll zu gliedern oder mit Kommentaren zu versehen.

2.4.4 Funktion

Leider konnte DieselTest in der Funktionalität nur wenig überzeugen - es eignet sich bestenfalls für kurze und sehr einfache Lasttests. Selbst unser relativ kleines Testszenario konnte nicht zur Zufriedenheit nachgestellt werden, da DieselTest offensichtlich mit dem Aufzeichnen von Cookies Probleme hat - obwohl laut Online-Hilfe das Programm mit Cookies umgehen können soll. Es war jedenfalls mit dem DieselTest-eigenen Browser nicht möglich, sich am Portal einzuloggen. Auch die ungewohnte, wenn auch übersichtliche Darstellung des Skripts in Tabellenform erwies sich im Test als eher umständlich und wenig komfortabel für nachträgliche Änderungen im Skript. Ein weiteres, großes Problem ist die fehlende Möglichkeit, aufgezeichnete Skripte noch einmal vor dem Lasttest auf Korrektheit zu prüfen, eine solche Funktionalität ist schlicht und einfach nicht vom Programm vorgesehen.

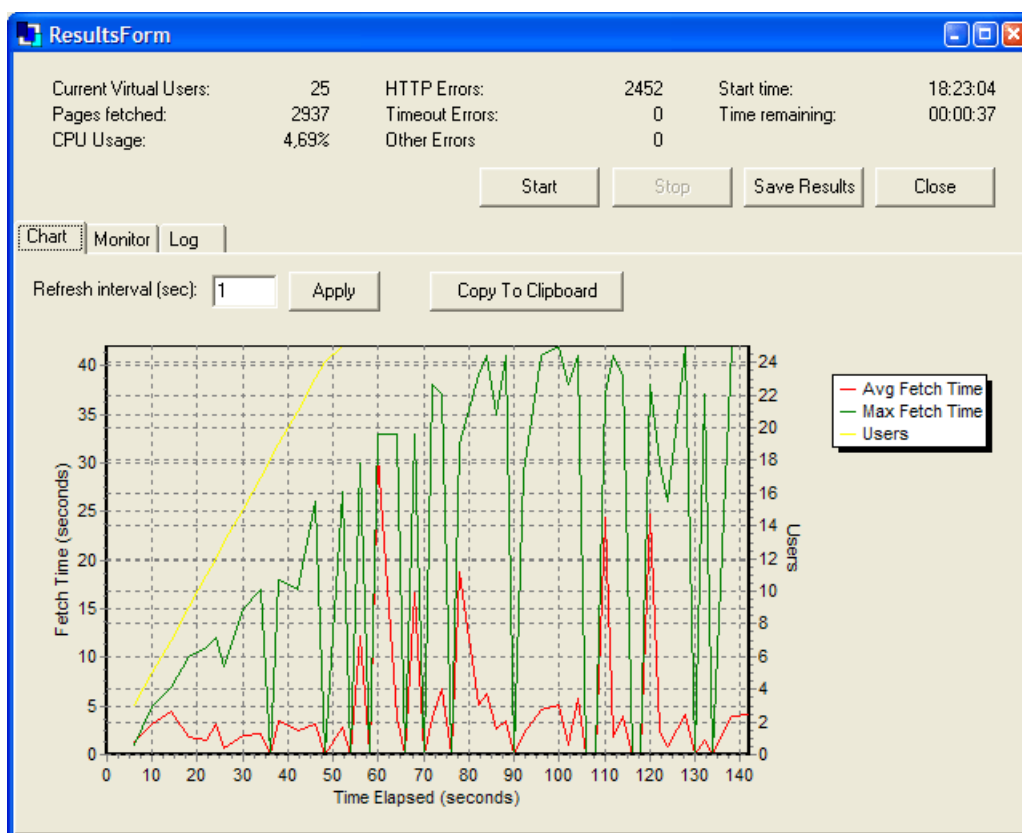


Abbildung 2.9: Testergebnisse und Auswertung mit Dieseltest

2.4.5 Sonstiges

Zum Support kann man eine klare Aussage treffen: Es gibt keinen! Durch die nicht mehr erreichbare Homepage war es nicht möglich, in irgendeiner Art an Support heranzukommen, sollte es Probleme geben, ist man auf sich alleine gestellt, da auch die Suchergebnisse über Google enttäuschend ausfielen.

2.4.6 Fazit

Dieseltest ist ein minimalistisches Lasttesttool, welches offensichtlich auch nicht mehr weiterentwickelt wird. Es ist daher nur sehr eingeschränkt zu empfehlen. Es ist durchaus geeignet, Traffic zu erzeugen, allerdings stört die mangelnde Cookie-Unterstützung, dadurch scheidet eine große Anzahl Testszenarien von vorne herein aus. Auch die mangelhaften Möglichkeiten der Testauswertung tragen dazu bei, dass Dieseltest eher zu den Schlusslichtern des Testfelds zu zählen ist.

2.5 Siege

Siege⁸ von *joedoc.org* wurde in der Version 2.59 getestet. Es bezeichnet sich selbst als HTTP-Test- und Benchmark-Werkzeug. Die Intention liegt darin, dass Webentwickler ihren fertigen Code unter Last setzen und so testen können. Das englische Wort „sieg“ bedeutet im Deutschen „Belagerung“. Und genau das macht das Werkzeug. Es versucht, wie die anderen Werkzeuge auch, mit vielen Benutzern gleichzeitig auf eine Webseite zuzugreifen. Das Tool unterstützt drei Modi: Den Regressionsmodus, in dem ein zuvor erzeugtes Webskript inkrementell abgearbeitet wird; den Internetmodus, in dem ein Internetadressen aus einem Webskript in zufälliger Reihenfolge aufgerufen werden und den Bruteforce Modus, in dem eine URL in der Kommandozeile übergeben und ohne Thinktimes getestet wird.

2.5.1 Voraussetzungen

Siege ist ein Unix Kommandozeilentool und hat daher geringe Systemvoraussetzungen. Ein Unix-System wird natürlich benötigt. Ansonsten wird in der Dokumentation nichts gefordert.

2.5.2 Technik

Solange das HTTP- oder HTTPS-Protokoll verwendet wird, kann Siege sowohl HTML- als auch Java-Anwendungen oder JSP-Seiten testen. Dabei werden auch Cookies angenommen und bis Sitzungsende gespeichert. Das Siege Tool ist ein C-Programm, das für Unix-Systeme geschrieben wurde. Laut Dokumentation kann es auf Solaris, Linux und HP-UX kompiliert und ausgeführt werden. Auf Windows-Plattformen ist das Werkzeug nicht einsetzbar. Kompiliert und getestet wurde es unter Linux. Spezielle Produkte kann Siege überhaupt nicht testen und auch bei der Protokollunterstützung sieht es nicht rosig aus. Die einzigen unterstützten Protokolle sind HTTP und HTTPS.

Über die Speicherauslastung pro Benutzers schweigt sich die Dokumentation aus, Tests ergaben jedoch, dass diese unter einem Megabyte liegt. Netzwerk-Monitorverfahren kennt Siege keine. Da man in Siege keine Skripte aufzeichnen kann, wird auch keines der

⁸Siege Homepage <http://www.joedog.org/siege>

Recording-Verfahren verwendet. Der Hersteller bietet allerdings einen Proxy-Server namens „SPProxy“ an, der besuchte Webseiten in einem Siegf-freundlichen Format ablegt, so dass man über dieses Zusatztool aufzeichnen kann. Die Funktionalität dieses Werkzeugs wurde allerdings nicht getestet, da es nicht zum Umfang von Siegf gehört. Insgesamt erhält Siegf in der Kategorie Technik eine Wertung von 26,1%.

2.5.3 Bedienbarkeit

Da Siegf nicht unter Windows einsetzbar ist, sind die Installationsfragen für Windows als irrelevant anzusehen. Unter Unix gibt es kein besonderes Shell-Script, welches die Installation durchführt. Hier kommt das gebräuchliche `./configure; ./make; ./make install` zum Einsatz. Dabei werden unter Linux keine Zusatzsoftware oder Patches benötigt, um das Tool zum Laufen zu bringen. Die Deinstallation wurde in der Dokumentation nicht erläutert. Vor der Verwendung muss man zuerst noch die Konfigurationsdatei mit einem Editor bearbeiten oder mit einem Script erzeugen. Der Administrationsaufwand während des Betriebs ist allerdings praktisch Null. Die Bedienung kann nicht als intuitiv bezeichnet werden, da sie komplett über Kommandozeilenparameter stattfindet. Eine schnelle Einarbeitung ist aber dadurch gewährleistet, dass das Werkzeug einen sehr beschränkten Funktionsumfang besitzt. Eine Dokumentation⁹ ist vorhanden. Allerdings war zum Testzeitpunkt der INSTALL-Bereich davon nicht verfügbar. Ein Tutorium existiert nicht, aber selbst die Dokumentation hat man in 10 Minuten durchgelesen und hat dann Kenntnis aller Möglichkeiten, die das Werkzeug besitzt. Als Online-Hilfe kann man die vorhandene Manpage sehen, die den gleichen Inhalt hat wie die Dokumentation. Abstürze kamen während der Tests nicht vor. Vor allem wegen der nicht vorhandenen Windows-Unterstützung und der etwas umständlichen und nicht erklärten Installation erhält das Siegf-Tool in der Kategorie „Bedienbarkeit“ lediglich 35,2%.

2.5.4 Funktion

Die Funktionalität von Siegf ist sehr eingeschränkt. Das Tool unterstützt von sich aus kein Script-Recording. Da man die Skripte allerdings selbst schreiben muss, sind sie später leicht nachvollziehbar. Algorithmus 2 zeigt ein manuell erstelltes Testskript für Siegf. Das ist aber auch nicht schwierig, denn es handelt sich um eine Aneinanderreihung von URLs, denen die GET oder POST Parameter übergeben wurden. Kommentare sind in den Skripten durch Unix-Typisches „#“ realisiert. Durch Zufall fanden wir beim Testen auch heraus, dass in den Skripten Systemvariablen (z.B. \$PATH) aufgelöst werden. Variablen können auch selbst gesetzt werden, was eine, wenn auch ziemlich eingeschränkte Parametrisierung ermöglicht. Da das Tool selbst keinerlei Funktionalität bietet, Skripte zu erzeugen, zu speichern und zu verwalten, wurden Fragen, die sich um die Skripte und die Skriptaufzeichnung drehen alle mit „Nein“ bewertet. Der Einsatz von Zufallswerten und Datentabellen wird ebenfalls nicht gewährleistet. Siegf unterstützt keine Form des Testmanagement. Auch verteilte Lasttests sind nicht zentral steuerbar und müssten auf jedem Client selbst gestartet und ausgewertet werden. Punkte für verteilte Lasttests können daher nicht gegeben werden. Überhaupt sind die einzigen Kriterien, die in der Unterkategorie „Testablauf“ noch positiv gewertet werden konnten: „Lasttest ins Internet“ und „Skript-Testlauf“, wobei ein Skript-Testlauf einfach dadurch besteht, dass ein Lauf mit einem Benutzer durchgeführt wird. Die Unterkategorie „Testauswertung“ sieht ähnlich leer aus. Die einzigen Punkte wurden hier für das Kriterium „Statistische Auswertung“ vergeben. Siegf misst die Gesamtzahl des erfolgten Datenverkehrs, die Antwortzeit des Servers, dessen Transaktionsrate, sein Durchsatz, die

⁹Siegf Dokumentation <http://www.joedog.org/siegf/docs/index.php>

Anzahl der gleichzeitigen Zugriffe und die Anzahl der positiven und negativen Antworten des Servers. Diese Daten werden für jeden Test in einem Logfile festgehalten. Messdaten über Hardware oder Betriebssysteme kann Siege nicht auswerten. Das Werkzeug erhält in der Kategorie „Funktionalität“ lediglich 14 Prozentpunkte.

Algorithm 2 Siege Testskript Professoren

```
# Professor Skript
# Variablen
ROOT=http://192.168.0.1/cms
# Download
#$(ROOT)
#$(ROOT)/tiki-file_galleries.php
#$(ROOT)/tiki-file_galleries.php?galleyId=3
#$(ROOT)/tiki-download_file.php?field=2
# Login
$(ROOT)/tiki-login.php POST user=prof&pass=prof&login=Anmeldung&stay_in_ssl.mode=
# Forum
# Hinweis: Es wird nur 1 Forumseintrag gesetzt, da Siege keine Parameter unterstuetzt.
$(ROOT)/tiki-forums.php
$(ROOT)/tiki-view_forum.php?forumId=7
$(ROOT)/tiki-view_forum.php POST comments_offset=0&comments_threadId=0&comments_threshold=0
&comments_sort_mode=commentDate_desc&forumId=7
&comments_postComment=Eintrag&comments_title=SiegeLoadtest
&comment_topicType=n&comments_data=EintragVomSiegeToolRandom=
# 5 Klicks
$(ROOT)/tiki-index.php?page=Professoren
$(ROOT)/tiki-index.php?page=Diplomarbeiten
$(ROOT)/tiki-index.php?page=Stundenplan
$(ROOT)/tiki-index.php?page=Mitarbeiter
$(ROOT)/tiki-index.php?page=Das%20Dekanat
# Logout
$(ROOT)/tiki-logout.php POST logout=Abmelden
```

2.5.5 Sonstiges

Was der Support angeht, so ist dieser wie bei so vielen Open-Source Werkzeugen eingeschränkt. Es gibt lediglich ein Email-Formular an die Entwickler. Da es noch immer neue Versionen des Werkzeugs gibt und auch vor kurzem erst der Zusatz Proxy-Server erschienen ist, wird davon ausgegangen, dass die Email-Anfragen in angemessener Zeit bearbeitet werden und die Community das entsprechende Know How besitzt, um bei Problemen zu helfen. Außerdem muss man dazu sagen, dass bei einem derart im Umfang eingeschränkten Produkt kaum Probleme auftreten sollten.

2.5.6 Fazit

Siege bezeichnet sich selbst als HTTP/HTTPS Stress Tester. Genau das ist es, nicht mehr und nicht weniger. Das Programm hat große Einschränkungen in sämtlichen getesteten Kategorien, kommt als einfaches Kommandozeilen-Tool daher und besitzt eine sehr beschränkte Testauswertung. Der einzige sinnvolle Anwendungsfall ist, einen Test durchzuführen, der herausfindet, ab wie vielen Benutzern bei einem bestimmten kleinen Skript ein Webserver zusammenbricht. Für größere Tests ist dieses Werkzeug allerdings ungeeig-

net und erhält in der Gesamtbewertung mit 985 von 4460 möglichen Punkten nur 21,1 Prozentpunkte.

Kapitel 3

Evaluationsergebnis

In diesem Kapitel werden die Ergebnisse der Evaluation analysiert. Dabei werden in Abschnitt 3.1 die Graphen, die das Evaluationstool automatisch erzeugt hat ausgewertet und verglichen. Man kann dadurch erkennen, in welchen Kategorien welche Werkzeuge die Oberhand haben. In Abschnitt 3.2 wird schließlich das Ergebnis erläutert, zu dem diese Evaluation geführt hat.

3.1 Vergleich

Abbildung 3.1 zeigt den Vergleich der Probanden in der Kategorie „Technik“. Hier wurden maximal 1340 Punkte vergeben. Im Vergleich zum Referenzsystem Loadrunner erreichen die Open-Source-Tools gerade einmal die Hälfte der Punkte. Das kommt vor allem durch die fehlende Protokoll- und Produktunterstützung der kostenlosen Werkzeuge. Im Vergleich untereinander liegen alle 5 Tools hier in etwa auf einem Level. JMeter und Testmaker können hier etwas Boden gutmachen, da beides Java-Programme sind und auf den verschiedensten Betriebssystemen laufen. Die zweite Kategorie, „Bedienbarkeit“ (siehe

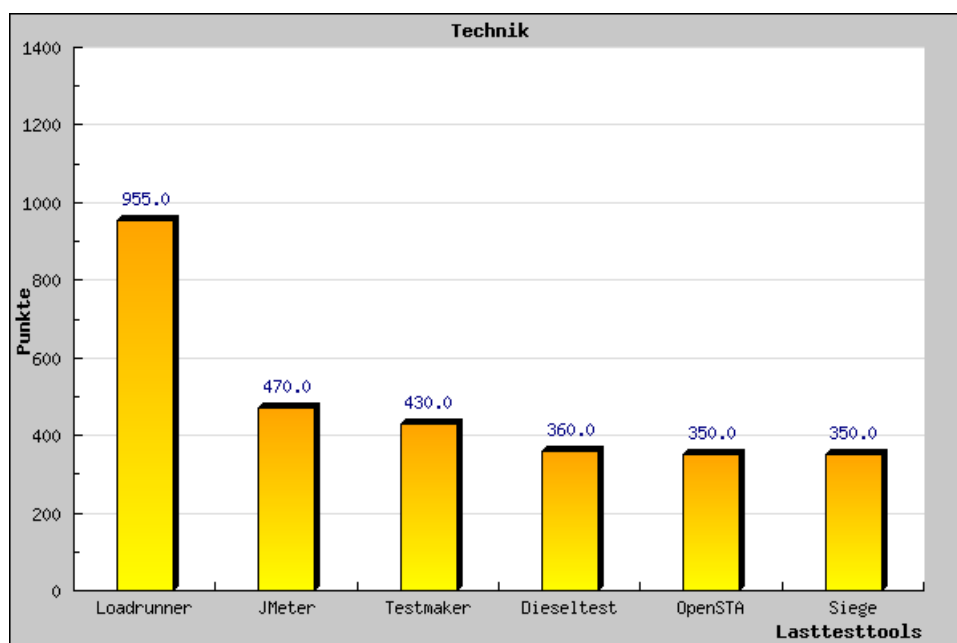


Abbildung 3.1: Lasttesttools - Technik

Abbildung 3.2) wurde mit maximal 610 Punkten gewertet. Hier können sich die meisten freien Werkzeuge vor dem Loadrunner behaupten, wobei man allerdings bedenken muss, dass die Bedienbarkeit einfacher wird, je geringer der Funktionsumfang eines Systems ist. Nur „Siege“ ist hier als Kommandozeilen-Werkzeug weit vom Rest abgeschlagen. Bei der

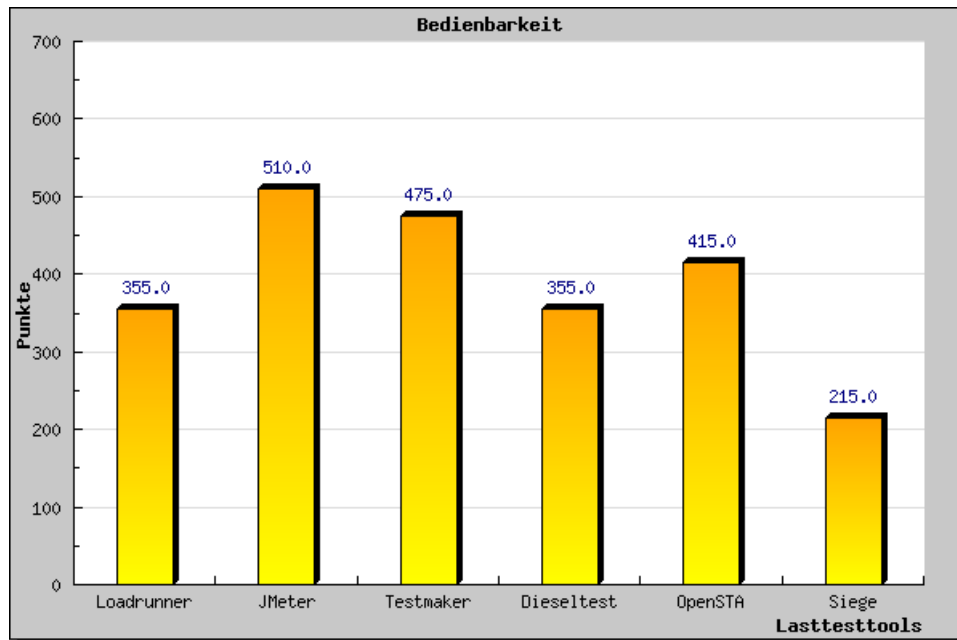


Abbildung 3.2: Lasttesttools - Bedienbarkeit

„Funktionalität“ ergeben sich, wie man in Abbildung 3.3 auf den ersten Blick sieht, immense Unterschiede. Der Loadrunner liegt hier mit 1890 von 2430 möglichen Punkten als Referenzsystem natürlich vorne, aber auch OpenSTA (1410) und JMeter (1240) haben einiges zu bieten. Testmaker konnte, wie bereits erläutert, in dieser Kategorie nicht vollständig getestet werden. Dieseltest und Siege haben einen zu geringen Funktionsumfang, um hier positiv abzuschneiden. Diese drei Werkzeuge sind in Sachen Funktionalität nur etwa halb so stark, wie die beiden erstgenannten. Abbildung 3.4 stellt den Vergleich des Supports dar. Da es sich bei den Probanden um Open-Source Software handelt kann der Support natürlich nicht in solchem Umfang erfolgen, wie beim Loadrunner. Sieger sind in dieser Kategorie Testmaker und OpenSTA. Siege und JMeter erreichen nur etwa die Hälfte dieser Tools, während die Dieseltest Homepage überhaupt nicht mehr erreichbar ist und somit in dieser Kategorie keine Wertung erhält. Sieht man sich nun den Gesamtvergleich (Abbildung 3.5) an, so reicht zwar keins der getesteten Werkzeuge an das Referenzsystem heran, das mit 3450 von 4660 Punkten eine Wertung von 74% erhält, aber sowohl OpenSTA (2300 = 49,4%), als auch JMeter (2295 = 49,2%) kommen dem Loadrunner in den getesteten Bereichen schon einigermaßen nahe. Im Mittelfeld ist mit 1605 Punkten Testmaker anzusiedeln. Das entspricht 34,4%. Dieseltest (1130 = 24,2%) und Siege (985 = 21,1%) liegen insgesamt im unteren Bereich, wobei Siege das Schlusslicht bildet. Abbildung 3.6 zeigt zum Abschluss noch die Verteilung der verschiedenen Werkzeuge an der Summe ihrer Wertungen. Daran erkennt man, dass Loadrunner, OpenSTA und JMeter sich 70% des Kuchen teilen, während Dieseltest, OpenSTA und Siege nur 30% der Gesamtpunktzahl beigetragen haben.

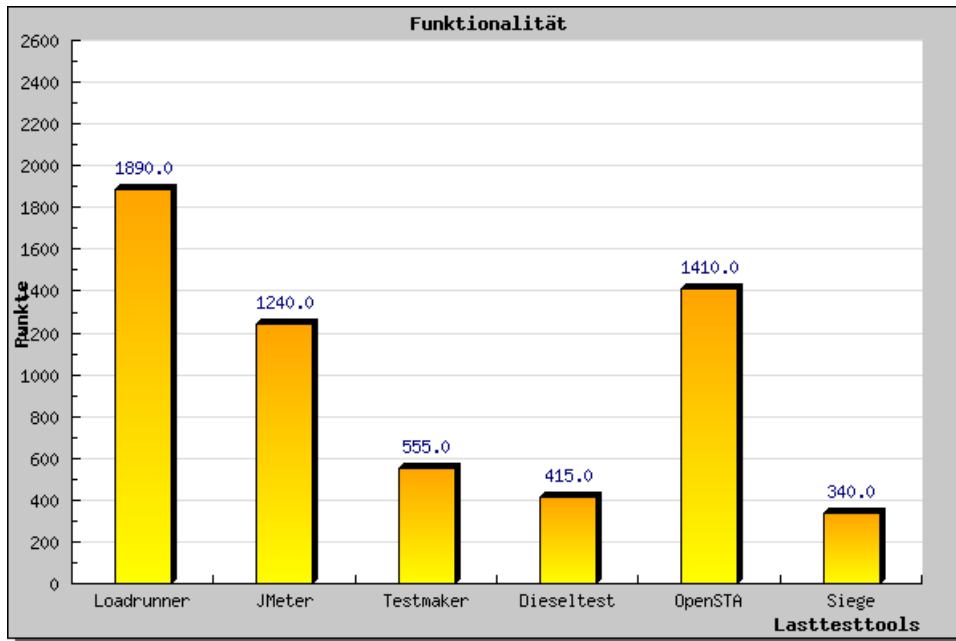


Abbildung 3.3: Lasttesttools - Funktionalität

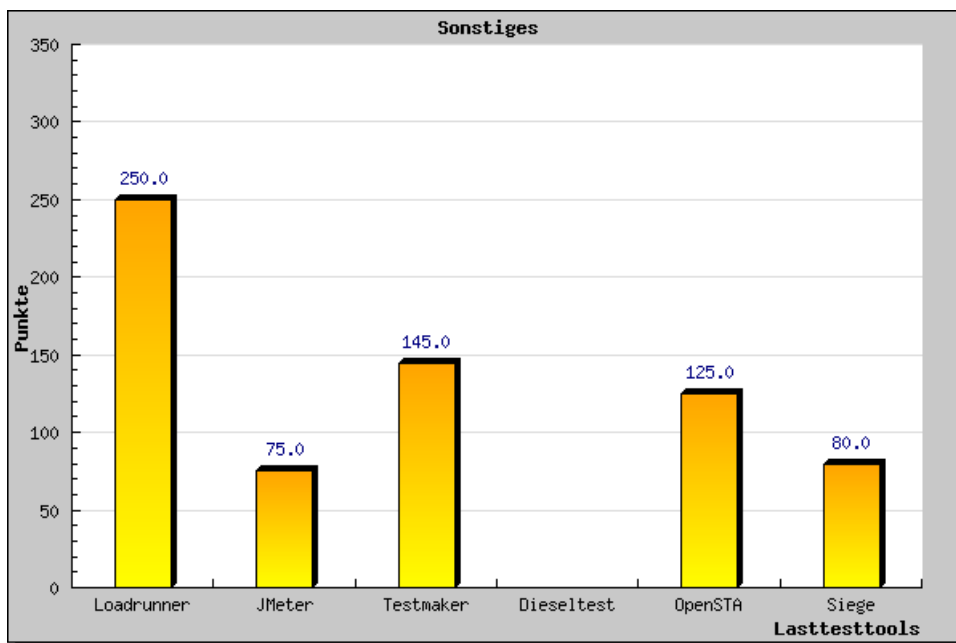


Abbildung 3.4: Lasttesttools - Sonstiges

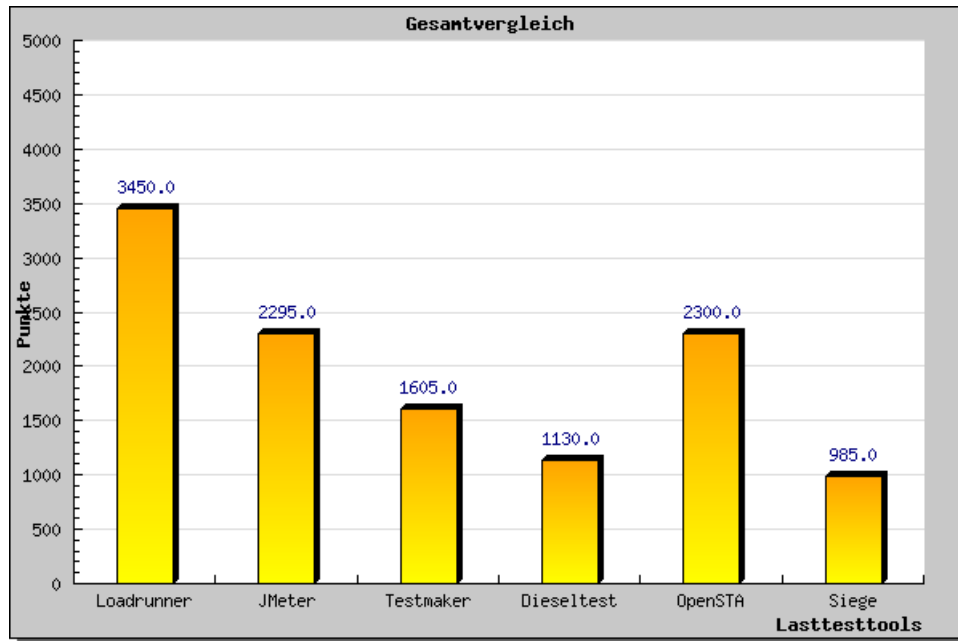


Abbildung 3.5: Lasttesttools - Gesamtvergleich

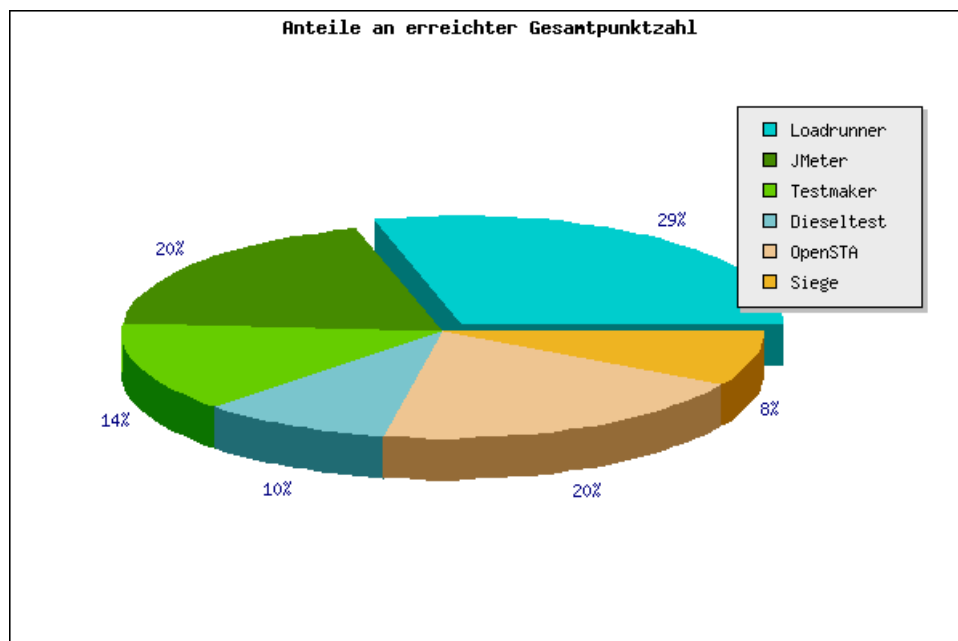


Abbildung 3.6: Lasttesttools - Verteilung

3.2 Fazit

Das Ergebnis der Evaluation teilt die Testteilnehmer grob in zwei Gruppen: Zum einen JMeter und OpenSTA, die in ihrem Funktionsumfang durchaus überzeugen konnten, und die anderen 3 Teilnehmer, die in aufgrund ihrer eingeschränkten Möglichkeiten wenn überhaupt nur in ganz speziellen Bereichen zu empfehlen sind. Sucht man ein plattformunabhängiges Lasttest-Tool und legt zudem Wert auf LDAP- und JDBC-Tests, so stellt sich der JMeter als erste Wahl heraus. Durch die Java-Technologie kann er auf beliebigen Systemen ausgeführt werden, einzige Voraussetzung ist eine lauffähige Java Virtual Machine. Sieht man einmal von der Plattformunabhängigkeit ab, so ist der Testsieger OpenSTA sicher das komfortableste und umfangreichste Produkt im Testfeld, wenn es um HTTP/HTTPS-Lasttests geht. Die reibungslos funktionierende Scriptaufzeichnung und -bearbeitung, die einfache Planung und Erstellung von Lasttests und auch die Möglichkeiten der Ergebnisauswertung machen OpenSTA zu einem mächtigen Werkzeug. Diesetest und Siege sind durchaus geeignet, Last zu erzeugen, jedoch muss man hier Abstriche in punkto Bedienkomfort und Ergebnisauswertung machen. Die Tatsache, dass die Diesetest-Homepage nicht mehr erreichbar ist, und die letzte Version aus dem Jahr 2001 stammt, zeigt zu dem, das dieses Produkt nicht mehr weitergepflegt wird - sollte man also ernsthaft über den Einsatz von Diesetest nachdenken, muss man in Betracht ziehen, dass man bei Problemen oder auftretenden Fehlern auf sich alleine gestellt ist. Siege ist sicher eine Alternative für Kommandozeilen-Fetischisten, jedoch muss man hier bedenken, dass gerade die wichtige Auswertung des Lasttests zu kurz kommt. Abschließend kann man daher sagen, das OpenSTA sich hervorragend auch für umfangreichere HTTP/HTTPS-Lasttests eignet, und auf diesem Gebiet durchaus eine (kostenlose) Konkurrenz zum teuren Branchenprimus Loadrunner sein kann.

Teil II

Performance-Studie einer Website

Kapitel 4

Aufbau der Testumgebung

4.1 Verwendete Hardware

Um bei den Performance-Tests Flaschenhals-Situationen so weit wie möglich zu vermeiden und realistische Vergleiche zu ermöglichen, führten wir sämtliche Tests in einer lokalen Testumgebung aus. Als Portalrechner und Webserver diente ein VIA Samuel 2 Prozessor mit 533 MHz und 256 MB Arbeitsspeicher. Dieser Rechner enthielt eine LAMP¹ Umgebung. Die verwendete Linux-Version war Mandrake Linux 9.1 mit einem Kernel der Version 2.4.21. Um die Tests auf dem Portalrechner durchzuführen wurden zwei Client-Rechner verwendet: Testclient 1 und Testclient 2. Testclient 1 war ein AMD Athlon XP 2200+ mit 512 MB Arbeitsspeicher. Das verwendete Betriebssystem war Microsoft Windows XP Professional. Auf dem Rechner wurde die Java Virtual Machine in der Version 1.4 installiert. Zusätzlich ein virtueller Rechner, um gegebenenfalls auch von diesem Tests fahren zu können. Als Testclient 2 fand ein Intel Pentium III mit 1000 MHz und 256 MB Arbeitsspeicher Verwendung. Auch auf diesem Rechner kam als Betriebssystem Microsoft Windows XP Professional zum Einsatz. Abbildung 4.1 zeigt den schematischen Aufbau der Testumgebung.

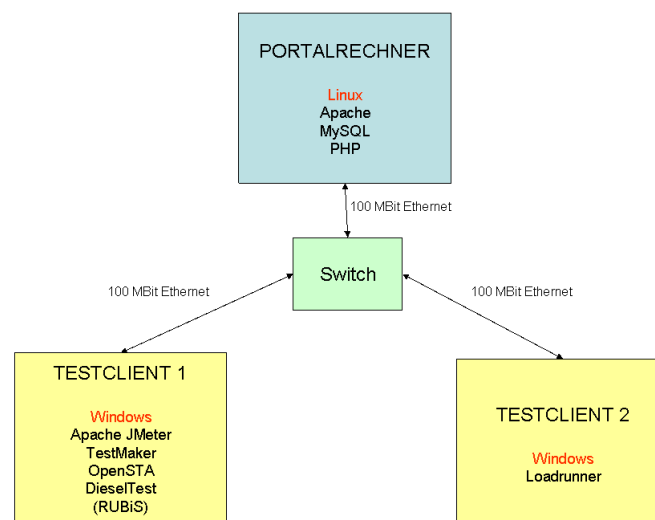


Abbildung 4.1: Schematischer Aufbau der Testumgebung

¹Linux, Apache, MySQL, PHP

Kapitel 5

Testszenario

Als Testszenario (siehe Abbildung 5.1) wählten wir das in der Veranstaltung *Open-Source-Portale* vom Team A1 erarbeitete Portal¹. Dieses verwendet als fertige Portalsoftware das Open-Source-Portal Tiki-Wiki. Das Testszenario wurde in zwei Skripte aufgeteilt. Das erste Testskript simuliert einen Professor, der zuerst im Downloadbereich den Grundlagenkatalog herunterlädt. Anschließend meldet er sich an und begibt sich in das Forum für Dozenten. Dort setzt er ein neues Posting ab. Anschließend führt er noch 5 Klicks zu statischen Seiten durch und meldet sich daraufhin wieder ab. Das zweite Skript simuliert

The screenshot shows the homepage of the MNI (Faculty of Mathematics, Natural Sciences, and Informatics) at FH-Giessen. The page layout includes a header with the university logo and navigation links for 'Mathematik', 'Naturwissenschaften', and 'Informatik'. On the left, there is a sidebar with an 'Anmeldung' (login) form, a search box, and an 'MNI Menü' (MNI Menu) with links to various sections like 'Hauptseite', 'Das Dekanat', 'Der Fachbereich', 'Professoren', 'Unternehmen', 'Labore', 'Mitarbeiter', 'Veranstaltungen', 'Homepages', 'Forum', and 'Download'. The main content area is titled 'Fachbereich MNI - FH-Giessen' and 'Willkommen im Fachbereich MNI'. It lists three study programs: 'Informatik als grundständiger Diplomstudiengang', 'der Aufbaustudiengang Strahlenschutz und -messtechnik und', and 'der Aufbaustudiengang Technische Redaktion & Multimediale Dokumentation'. A photo of Prof. Dr.-Ing. Axel Schumann-Luck, Dean of the MNI faculty, is shown. The footer includes 'Erstellt von: system letzte Änderung: Sonntag 07 of Dezember, 2003 [18:32:21 UTC] von mm_admin', 'Historie exportieren', and various technology logos like 'MADE WITH CASCADING STYLE SHEETS', 'W3C XHTML 1.0', 'powered by pear', 'powered by php', and 'smarty'.

Abbildung 5.1: Startseite unseres Testportals

einen Studenten. Dieser versucht sich gleich anzumelden, trägt jedoch zuerst ein falsches Passwort ein. Anschließend meldet er sich richtig an. Ist er angemeldet, geht der Student

¹despina.mni.fh-giessen.de/a1/cms

auf die Professorenseite, sieht sich die Druckansicht an und begibt sich wieder zurück. Auch der Student besucht nun das Forum. Allerdings den Web Performance Bereich. Dort antwortet er auf ein bereits vorhandenes Thema und lässt sich zuvor die Vorschau seines Eintrags anzeigen. Anschließend wechselt er zum Webmail-Bereich. Da das Testsystem nicht für Mail konfiguriert ist, erhält er eine Fehlermeldung. Diese übergeht er mit einem Klick auf die Einstellungen und legt einen neuen Mail-Account an. Danach führt auch der Student fünf Klicks auf statische Seiten aus und meldet sich ab. Der Aufbau der beiden Skripte befindet sich mit genaueren Infos zu den Eingaben in Listenform im Anhang. Bei ersten Tests fiel auf, dass bei der Verwendung von mehreren Benutzern immer nur der erste einen Forumeintrag hinterließ. Das liegt daran, dass das CMS² keine identischen Foreneinträge erlaubt. Beachtet werden muss bei weiteren Tests also, dass man Einträge im Forum in irgendeiner Art parametrisiert. Dazu eignet sich bspw. das Eintragen eines Zufallswerts oder eines Zeitstempels. Die Tests sind so vorgesehen, dass beide Skripte, sofern es die Testtools zulassen, parallel gestartet werden. Dabei sollen zehn Studenten mit 2 Professoren konkurrieren.

²CMS - Contend Managment System

Kapitel 6

Testdurchführung

6.1 Mercury Loadrunner

Da das Testszenario mittels des Loadrunners erstellt wurde, führt dieser alle Tests natürlich wie gewünscht aus. Das Problem mit den identischen Forumseinträgen wurde im Loadrunner so gelöst, dass der Quellcode des aufgezeichneten Skripts editiert wurde (Code siehe Algorithmus 3). Dabei wurde eine neue Funktion geschrieben, die nun bei jedem Skriptdurchlauf an der Stelle aufgerufen wird, an der zuvor der feste Forumstext stand. Die Funktion gibt nun den festen Forumstext zurück, hängt aber an dessen Ende noch eine Zufallszahl an. Dadurch sind die Foreneinträge nicht mehr identisch und das Forum übernimmt alle Einträge. Die Zufallszahl wurde durch die C-Funktion `rand()` erzeugt, die in der Loadrunner Skriptumgebung enthalten ist. Was die Auswertung angeht, so wurden

Algorithm 3 Loadrunner Foreneintragsfunktion

```
extern char* GetForumText();
extern char *strcat ( char *to, const char *from );
extern char *itoa ( int value, char *str, int radix );
extern char* GetForumText()
{
    //Zufallszahl für Forumsseintrag erzeugen
    int r = rand();
    static char cr[50];
    static char cValue[4096];
    //char ia[10];
    char *ia = itoa(r, cr, 10);
    static char *rval;
    lr_output_message(":TESTAUSGABE: cr = %s", cr);
    strcat(cValue, "Value=Dies ist ein Neues Posting, Random=");
    lr_output_message(":TESTAUSGABE: cValue = %s", cValue);
    strcat(cValue, cr);
    lr_output_message(":TESTAUSGABE: cValue = %s", cValue);
    return cValue;
}
```

mit dem Loadrunner zwei Durchläufe mit Ramp-Up erzeugt. Zuerst mit 10 Studenten und 2 Professoren. Danach mit 20 Studenten und 4 Professoren. Beim Loadrunner wurde nachträglich das Testszenario etwas abgeändert. Es wurde sowohl für das Professoren-, als auch für das Studentenskript variable Datenlisten erzeugt, so dass sich die Login-

Namen bei jedem Durchlauf ändern. Die folgenden Abbildungen zeigen einige Ergebnisse des Loadrunner Tests. Dabei zeigt der obere Teil der Grafik immer das Szenario mit 10/2 Benutzern und der untere Teil das gleiche Szenario mit 20/4 Benutzern. An Abbildung 6.1

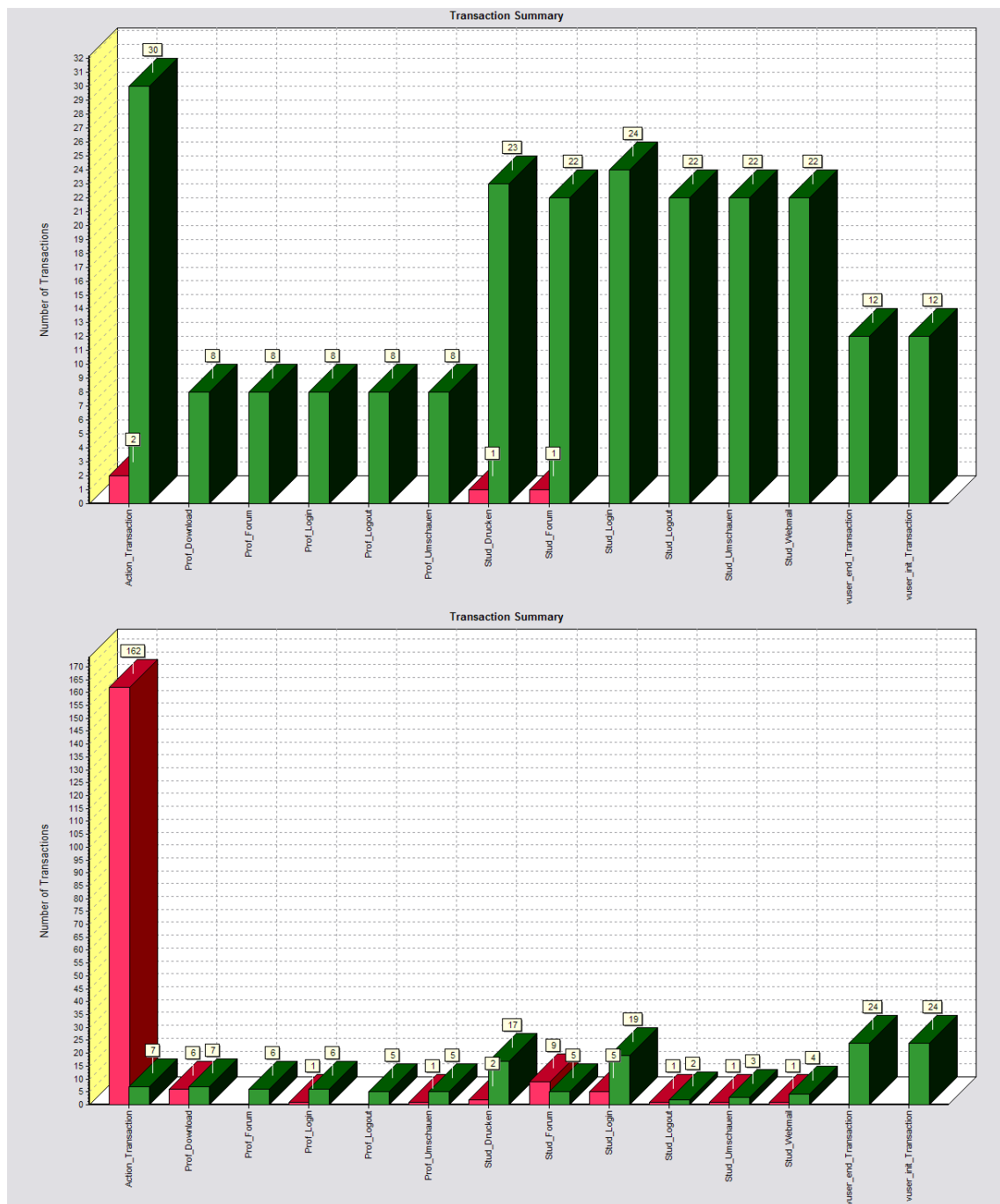


Abbildung 6.1: Loadrunner - Transaktionsvergleich

erkennt man deutlich, dass die Hardware für 24 gleichzeitige Benutzer zu schwach ausgelegt ist. Sind bei 12 Benutzern nahezu alle Tests durchgelaufen, so kam bei 24 Benutzern das System sozusagen zum Stillstand. Dies wird auch an den Durchsatzgraphen deutlich, die in Abbildung 6.2 dargestellt werden. Der Durchsatz im oberen Graphen variiert sehr stark. Das kommt vermutlich daher, dass zwischen jeder Anfrage Thinktimes liegen, die

ein natürliches Surfverhalten simulieren sollen. Je nachdem ob gerade viele oder wenige virtueller Benutzer eine Thinktime abwarten, steigt oder fällt der Durchsatz. Im unteren Graph sieht man, dass am Ende des Tests der Server-Rechner dicht macht und der Durchsatz praktisch bei Null ist. Eine Erklärung dafür kann auch die Auswertung des Siege-Tools



Abbildung 6.2: Loadrunner - Durchsatz

in Abschnitt auf Seite 40 aufzeigen. Zum Abschluss der Loadrunner-Auswertung in diesem Dokument wird noch eine Übersicht der beiden Szenarien gezeigt. Um genauere Vergleiche zu ziehen, verwende man die beiliegenden Analyseresultate für das Loadrunner Analyseprogramm.

- **Maximum Running Vusers:** 13
- **Total Throughput (bytes):** 24.046.493
- **Average Throughput (bytes/second):** 15.554
- **Total Hits:** 2.312
- **Average Hits per Second:** 1,495 [View HTTP Responses Summary](#)

Transaction Summary

- **Transactions:** Total Passed: 229 Total Failed: 4 Total Stopped: 0 [Average Response Time](#)

Transaction Name	Minimum	Average	Maximum	Std. Deviation	90 Percent	Pass	Fail	Stop
Action Transaction	110,41	482,776	623,088	139,955	618,36	30	2	0
Prof Download	9,342	36,572	51,49	16,163	51,479	8	0	0
Prof Forum	21,526	60,605	78,193	19,525	78,188	8	0	0
Prof Login	5,629	29,375	40,847	12,65	40,844	8	0	0
Prof Logout	13,176	29,965	37	9,365	36,996	8	0	0
Prof Umschauen	43,428	107,316	132,686	31,78	132,671	8	0	0
Stud Drucken	23,353	79,842	94,746	18,922	92,643	23	1	0
Stud Forum	84,518	173,489	197,521	31,93	194,364	22	1	0
Stud Login	12,217	54,295	67,796	14,776	65,843	24	0	0
Stud Logout	3,501	26,108	37,027	11,653	36,313	22	0	0
Stud Umschauen	19,487	76,046	96,367	23,755	95,985	22	0	0
Stud Webmail	13,818	58,3	72,413	16,949	70,297	22	0	0
vuser_end Transaction	0	0	0	0	0	12	0	0
vuser_init Transaction	0,001	0,001	0,001	0	0	12	0	0

- **Maximum Running Vusers:** 25
- **Total Throughput (bytes):** 15.287.978
- **Average Throughput (bytes/second):** 7.191
- **Total Hits:** 1.591
- **Average Hits per Second:** 0,748 [View HTTP Responses Summary](#)

Transaction Summary

- **Transactions:** Total Passed: 134 Total Failed: 189 Total Stopped: 0 [Average Response Time](#)

Transaction Name	Minimum	Average	Maximum	Std. Deviation	90 Percent	Pass	Fail	Stop
Action Transaction	103,046	324,42	489,774	127,197	489,74	7	162	0
Prof Download	7,256	37,542	85,38	24,286	85,367	7	6	0
Prof Forum	23,498	66,842	135,862	37,046	135,857	6	0	0
Prof Login	7,364	24,961	45,156	12,266	45,148	6	1	0
Prof Logout	11,51	43,902	65,955	19,982	65,943	5	0	0
Prof Umschauen	38,416	124,132	192,902	54,447	192,896	5	1	0
Stud Drucken	22,78	99,552	141,181	34,589	140,525	17	2	0
Stud Forum	91,499	202,47	341,266	89,119	341,243	5	9	0
Stud Login	12,189	80,832	158,533	38,703	148,52	19	5	0
Stud Logout	25,803	37,917	50,032	12,114	50,017	2	1	0
Stud Umschauen	65,628	112,643	164,295	40,414	164,283	3	1	0
Stud Webmail	35,416	85,148	137,566	37,955	137,561	4	1	0
vuser_end Transaction	0	0	0	0	0	24	0	0
vuser_init Transaction	0,001	0,001	0,001	0	0	24	0	0

Abbildung 6.3: Loadrunner - Übersicht

6.2 Apache JMeter

Um das vorgegebene Szenario zu realisieren, wurde zunächst das Professoren- und dann das Studentenskript über den Proxy erzeugt. Dies gestaltete sich etwas schwierig, da es nur umständlich möglich ist, beim JMeter während des Aufzeichnens Transaktionen zu setzen. Getestet wurde das gleiche Szenario wie beim Loadrunner. 10 Studenten und 2 Professoren setzten das CMS unter Last. Allerdings ohne den Zusatz der variablen Logins, obwohl auch dies möglich gewesen wäre. Das Ergebnis des durchgeführten Tests fällt leider etwas mager aus, da JMeter, wie bereits bei der Evaluation festgestellt, eine ungenügende Datenauswertung bietet. Abbildung auf Seite 11 zeigt übrigens bereits das Testergebnis für das durchgeführte Testszenario. Erkenntnisse lassen sich daraus nicht ziehen, außer vielleicht, dass dringend neue JMeter Listener programmiert werden müssten. Die Tabellenauswertung (Abbildung 6.4) ist jedoch ganz brauchbar. Dort werden die Anzahl der Aufrufe, die minimalen, maximalen und durchschnittlichen Aufrufzeiten und die fehlerhaften Aufrufe in Prozent dargestellt. Die fehlerhaften Aufrufe wurden von uns gelb markiert. Dabei

URL	Count	Average	Min	Max	Error%	Rate
/cms	20	26	0	173	0,00%	2//hour
/cms/lib/tiki-js.js	827	517	0	17487	0,00%	7,8//hour
/cms/	90	11446	1407	15862	0,00%	8//hour
/cms/tiki-index.php	467	19031	3397	39242	0,00%	4,4//hour
/cms/styles/mni.css	733	16	0	1453	0,00%	6,9//hour
/cms/tiki-file_galleries.php	38	22	0	219	50,00%	4//hour
/cms/tiki-login.php	130	12721	3116	17332	46,15%	1,2//hour
/cms/tiki-list_file_gallery.php	28	16721	5813	21738	0,00%	3//hour
/cms/tiki-error.php	59	12989	3375	16406	0,00%	6//hour
/cms/tiki-download_file.php	15	7722	6111	9957	0,00%	1//hour
/cms/tiki-forums.php	52	19049	14862	24505	0,00%	1,5//min
/cms/tiki-view_forum.php	64	20316	14440	27473	0,00%	1,9//min
/cms/styles/mni.css/cms/lib/tiki-js.js	36	16	0	78	0,00%	1,1//min
	156	1814	0	22785	61,54%	4,3//min
/cms/tiki-print.php	40	32008	27379	35382	0,00%	1,4//min
/cms/lib/tiki-js.js/cms/lib/tiki-js.js	24	77	0	1469	0,00%	42,8//hour
/cms/tiki-my_tiki.php/cms/styles/mni.css	24	734	359	2735	100,00%	42,9//hour
/cms/lib/tabs/utills.js	12	45	0	390	0,00%	21,4//hour
/cms/lib/tabs/global.js/cms/lib/tabs/viewport.js	12	526	360	812	100,00%	21,3//hour
/cms/lib/tabs/tabs.js/cms/lib/tabs/cookie.js	24	12	0	47	0,00%	42,6//hour
/cms/tiki-logout.php	42	12919	8986	15893	0,00%	1,2//min
/cms/tiki-view_forum_thread.php	117	24203	11517	31677	0,00%	3,8//min
/cms/tiki-webmail.php	61	15717	7548	20035	0,00%	2,0//min
/cms/img/wiki_up/mailto.gif	30	60	0	719	0,00%	1,4//min
/cms/lib/tiki-js.js/cms/styles/mni.css	60	42	0	1703	0,00%	2,8//min
TOTAL	3161	6827	0	39242	6,68%	29,7//hour

Abbildung 6.4: JMeter - Tabellenauswertung

fällt besonders das Login auf, das mit 46,15% aller Aufrufe fehlgeschlug. Die File-Galleries mit genau 50,00 % könnten wegen der auffällig geraden Zahl auch durch einen Fehler im Testskript fehlgeschlagen sein. Leider geben die übrigen Lauscher nicht viel her und das Öffnen der Testergebnisse in Excel funktionierte aufgrund des XML-Formates ohne Probleme und man erhielt eine übersichtliche Tabelle mit Filtern, jedoch konnte man auch daraus nicht erkennen, woraus die Fehler beim Login resultierten. Man kann lediglich ermitteln, dass der Aufruf der login.php in einem HTTP-Fehlercode 203 resultiert. Dies ist jedoch nichts ungewöhnliches und sowohl bei allen fehlgeschlagenen als auch erfolgreichen Aufrufen der Fall. Code 203 bedeutet lediglich, dass in einem Request keine Datei sondern ein Verzeichnis angegeben wurde und an eine Standard-Datei weitergeleitet wird. Im Falle der login.php bedeutet dies nur, dass nach dem Login die index.php aufgerufen wird, die Standard-Datei in diesem Verzeichnis. Die Test-Skripte sowie das Analysefile befinden sich bei den beigefügten Skript-Dateien.

6.3 Testmaker

Mit dem Testmaker konnte kein Lasttest durchgeführt werden. Für genaue Erläuterung der Gründe siehe Kapitel auf Seite 12. Die Testmaker Skriptdateien befinden sich zum selber testen natürlich auch in der Skriptsammlung.

6.4 OpenSTA

Zur Durchführung des Testes wurden zunächst die beiden Testszenarien mit dem Script Modeler von OpenSTA aufgezeichnet. Um den Zufallswert, der für die Einträge im Forum benötigt wird, zu erzeugen, wurde mit Hilfe des Assistenten des Script Modelers eine neue Variable angelegt und 10000 zufällige Werte erzeugt. Diese Variable wurde dann im Skript an der Stelle, an dem die Eintragung im Forum vorgenommen wird, eingefügt. Anschließend wurde mit dem OpenSTA Commander ein Lasttest vorbereitet. Hierzu wurden die beiden erzeugten Skripte (Professoren und Studenten) verwendet. Der Test wurde auf 5 Professoren und 10 Studenten beschränkt, da beim ersten Test mit dem Loadrunner mit 25 VUs das Portal bereits den Dienst quittiert hatte. Die VUs wurden mit Hilfe eines Ramp-Ups so konfiguriert, dass nach zwei Minuten die volle Last von 15 VUs erzeugt wurde. Es sollten zwei komplette Durchläufe der Skripte komplettiert werden. Die Testdurchführung bereitete keinerlei Probleme. Der Test dauerte etwa 15 Minuten, und wurde ohne Fehler beendet. Anschließend wurden die Testergebnisse in Excel exportiert, um sie besser auswerten zu können.

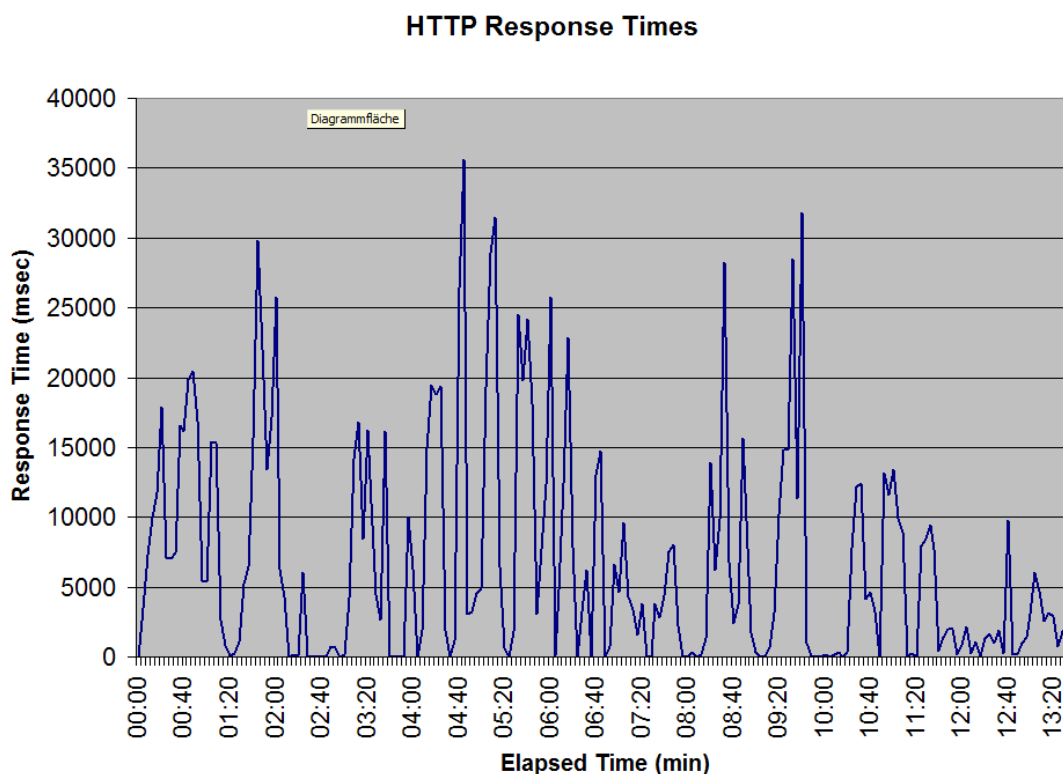


Abbildung 6.5: OpenSTA - Response-Times

6.5 DieselTest

Da mit DieselTest die von uns festgelegten Testszenarien aufgrund der mangelnden Cookie-Unterstützung nicht durchführbar waren, entschieden wir uns, ein einfacheres Testszenario, bei dem lediglich auf der Hauptseite ohne Benutzeranmeldung etwa 10 Klicks auf verschiedene Bereiche getätigt wurden, zu erstellen. Dies klappte wiederum problemlos, und dieser vereinfachte Test wurde wieder mit 25 VUs durchgeführt, immerhin wurden keine ressourcenschluckenden Einträge im Forum vorgenommen. Die Testdauer wurde auf fünf Minuten begrenzt, ein Ramp-Up von einer Minute für alle VUs wurde ebenfalls eingestellt. Der Test wurde mit Fehlern abgeschlossen, am Ende zeigte DieselTest 2245 HTTP-Errors an. Aufgrund der mangelhaften Auswertungsmöglichkeiten und der fehlenden Möglichkeit, das Skript überhaupt auf Korrektheit zu testen, besitzt dieser Test jedoch wenig Aussagekraft.

6.6 Siege

Der Siege Lasttest verlief erwartungsgemäß sehr spärlich. Mit diesem Tool konnte sehr simpel herausgefunden werden, wie viele Benutzer auf der Testumgebung maximal gleichzeitig agieren können. Dazu benötigte man keinen sehr großen Aufwand. Lediglich das Professoren-Testskript aus Abbildung 2 auf Seite 23 musste manuell erzeugt werden. Da Siege keine zwei Skripte gleichzeitig ausführen kann, wurde lediglich die maximale Anzahl von gleichzeitig vorhandenen Professoren ermittelt. Es wurde mit einem Benutzer begonnen. Die Anzahl wurde in 5er Schritten erhöht. Jede Stufe wurde 5 Minuten lang ausgeführt. Als Testergebnis konnte folgendes festgehalten werden.

User	Treffer	Verfügbarkeit (%)	Datentransfer (byte)	Antwortzeit (Sek)	Transakt.-Rate (Tr/Sek)	Durchsatz (byte/Sek)	Konkurrenz	Erfolg	Fehler
1	70	100	16030	3,61	0,23	53,37	0,84	63	7
5	120	100	27210	11,89	0,40	90,70	4,76	105	15
10	130	100	29140	21,39	0,43	97,12	9,27	110	20
15	120	100	27210	33,58	0,40	90,59	13,42	105	15
20	97	100	21286	53,27	0,32	70,86	17,20	77	20
25	61	100	12628	56,78	0,20	42,14	11,56	40	21

Die Ergebnisse wurden in Excel in einem Graphen ausgewertet. An Abbildung 6.6 kann man sehr schön erkennen, dass die beste Performance bei etwa 10 gleichzeitigen Benutzern liegt. Dort haben die Anzahl der Treffer während der 5 Minuten und der Datendurchsatz ihr Maximum. Bei weniger Benutzern liegen diese Werte darunter, da die Transaktionsrate, also die Transaktionen pro Sekunde, im Grunde recht konstant zwischen 0,2 und 0,4 liegen. Greift also nur ein Benutzer auf den Webserver zu, muss er sich zwar keine Ressourcen teilen, aber eine Transaktion dauert fast genauso lange wie sonst. Außerdem ist der Durchsatz bei einem Benutzer natürlich viel geringer als bei 10 Benutzern. Testet man nun mit 15 oder mehr Benutzern, so gehen sämtliche Raten ziemlich in den Keller. Das liegt vor allem daran, dass auf dem Server-Rechner nur 256 MB Arbeitsspeicher zur Verfügung stehen und dieser vor allem damit beschäftigt ist, ungenutzte Daten auf die Festplatte auszulagern oder benötigte von dort zu beschaffen. Abbildung 6.7 zeigt Systeminformationen des Serverrechners bei 25 gleichzeitigen Benutzern. Dort kann man erkennen, dass die CPU nur zu 4% ausgelastet ist, während 243328 kByte des 247664 kByte

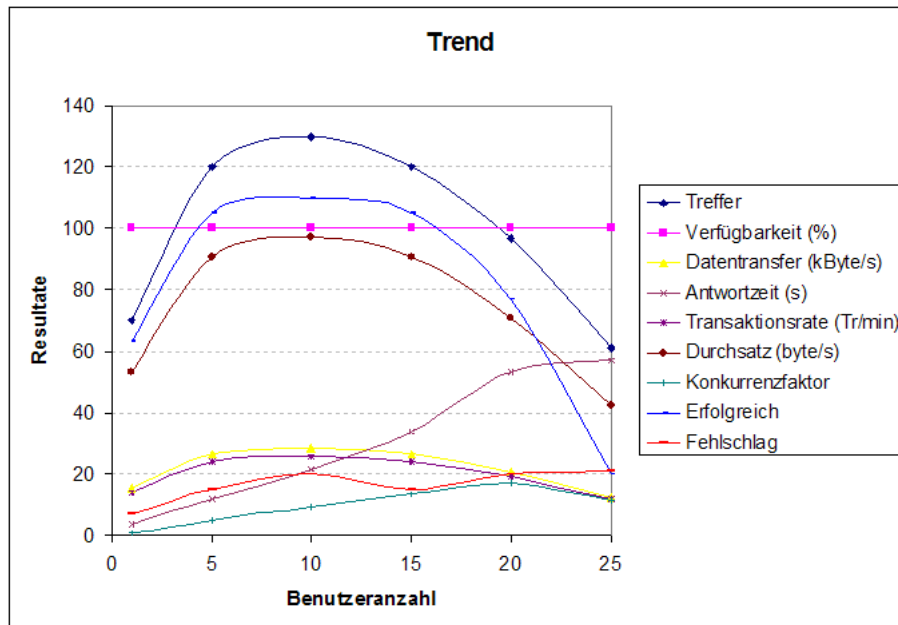


Abbildung 6.6: Auswertung der Siege Tests

großen Arbeitsspeichers verbraucht sind. Bei 25 und 30 Benutzern war das Festplattenled im Dauerbetrieb. Das schlechte Abschneiden bei dieser Anzahl von Nutzern kann man daher nicht dem Webserver oder dem darauf befindlichen Portal anlasten, sondern der ungenügend ausgestatteten Testumgebung.

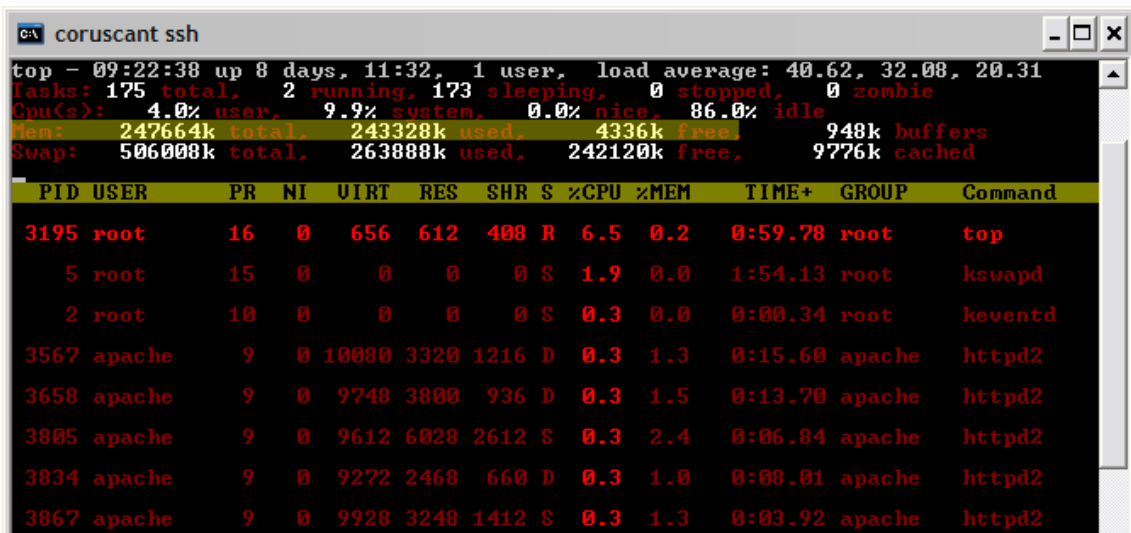


Abbildung 6.7: Siege - Systemauslastung bei 25 Benutzern

Kapitel 7

Testergebnis und Verbesserungsvorschlag

Mit den Werkzeugen Loadrunner, JMeter, OpenSTA und Siege ließ sich unser zuvor festgelegtes Szenario testen. Diesetest und Testmaker scheiterten bereits am Login. Aus den erfolgreichen Tests der anderen Werkzeuge lassen sich einige Schlüsse ziehen. Beim Erstellen der Tests zeigte sich gleich, dass Loadtesttools, die nicht mit Parametern arbeiten können, bei dynamischen Seiten wenig Chancen haben. Siege z.B. konnte mangels Parametrisierung keine Foreneinträge im getesteten CMS erzeugen. Die unterschiedlichen Lasttests konnten einige Ergebnisse aufweisen. Diese haben allerdings wenig Aussagekraft bezüglich der verwendeten Web-Software, sondern zeigen vielmehr Schwächen in der Hardware des Server-Rechners auf. Die zuerst ausgeführten Tests mit dem Loadrunner zeigen, dass es keinerlei Probleme bei 12 parallel agierenden Benutzern auf dem Zielsystem gibt. In der Loadrunner-Analyse, die natürlich auch beigelegt ist, erkennt man, dass die Transaktion „Stud_Forum“ die längste Zeit beansprucht. Der Grund dafür müsste noch in weitergehenden Analysen verfolgt werden. Diese Transaktion ähnelt nämlich der Transaktion „Prof_Forum“, die von wesentlich kürzerer Dauer ist. Der einzige Unterschied besteht dabei darin, dass in der „Stud_Forum“ Transaktion noch eine Vorschau enthalten ist und kein neuer Thread erstellt, sondern auf einen bestehenden geantwortet wird.

Abbildung 6.2 auf Seite 36 zeigt, dass beim zweiten Testlauf, mit 24 Benutzern spätestens ab Minute 24:00 der Server komplett dicht macht. Was man nur in den Analysedaten des Loadrunner sehen kann, ist, dass ab Minute 10:40 bereits alle 24 virtuellen Benutzer aktiv waren. Auch das Ramp-Down ab Minute 28:00 konnte nicht bewirken, dass die übrigen Benutzer wieder Durchsatz erzeugen konnten. Erst ca. 15 Minuten nach Ende des Tests war der Server wieder erreichbar. Auch SSH und andere Dienste konnten während dieser Zeitspanne nicht erreicht werden. Die LED der Fesplatte stand auf Dauerbetrieb. Dieser Stresstest zeigt zum einen, dass Performance-Tests über einen längeren Zeitraum laufen müssen, denn Siege hatte bei 25 Benutzern nach fünf Minuten noch immer eine Verfügbarkeit von 100%. Zum anderen zeigt der Test, dass der verwendete Server bereits bei einer Anzahl von 24 Daueranfragen einer DOS-Attacke¹ ausgesetzt werden kann. Auch die mit Excel ausgewerteten Response-Times des Werkzeugs OpenSTA (Abbildung 6.5 auf Seite 39) unterstrichen hier, dass bereits bei 15 gleichzeitigen Benutzern kein gleichmäßiges Bedienen der Anfragen gewährleistet werden konnte. Mit dem bei der Evaluation als so schlecht eingestuften Werkzeug Siege konnten hier jedoch einige sehr gute Ergebnisse erzielt werden. Bei der schrittweisen Erhöhung der Benutzer um jeweils 5 zeigt Abbildung auf Seite 40 aufbauend auf die Ergebnisse des Loadrunners deutlich, wo die Probleme des Testservers

¹DOS - Denial Of Service (Dienstverweigerung)

beginnen. Ab etwa 15 parallelen Benutzern gehen Treffer und Durchsatz drastisch nach unten, wobei die Antwortzeiten gleichzeitig stark anstiegen. Mit dem Einsatz des Unix Werkzeugs `top` konnte das Problem schnell isoliert werden. Der Flaschenhals auf unserem Testserver ist ganz klar der Arbeitsspeicher. Bei nur 4MB freiem Arbeitsspeicher bei 25 virtuellen Benutzern war der Rechner nur noch damit beschäftigt, Daten vom Speicher auf die Festplatte auszulagern und benötigte Daten in den Speicher zu schreiben. Die CPU-Auslastung von 4% zeigt dies sehr deutlich, da sie durch UDMA an diesen Vorgängen nicht beteiligt war und im Grunde nicht genutzt werden konnte. Als Ergebnis der Tests lässt sich also sagen, dass 256 MB Arbeitsspeicher bei der CPU des Testsystems für maximal 10 gleichzeitige permanente Benutzer ausreichend sind. Möchte man die Verfügbarkeit und Performance bei noch mehr Benutzern gewährleisten, muss unbedingt der Arbeitsspeicher erweitert werden. Anschließend müssen neue Tests durchgeführt werden.

Anhang A

Evaluation der Testwerkzeuge

A.1 Evaluationsfragen

Diese Liste zeigt die Evaluationsfragen, die in dem Testtool beantwortet werden mussten. Die Zahlen in Eckigen Klammern sind die maximale Punktzahl für die entsprechende Frage. Diese richtet sich nach der Gewichtung der Excel-Vorlage.

A.1.1 Technik

Testumfeld

- Wurde auf Java 2 (JDK 1.3/1.4) getestet? [50]
- Wurde auf HTML/Javascript getestet? [50]

Betriebssysteme

- Ist eine WindowsNT/[20]00/XP Version verfügbar? [50]
- Ist eine Solaris Version verfügbar? [50]
- Ist eine Linux Version verfügbar? [50]
- Ist eine HP-UX Version verfügbar? [20]
- Ist eine IBM S/390 Version verfügbar? [10]
- Ist eine Tandem Version verfügbar? [10]

Können folgende Produkte gemessen werden?

- Apache? [40]
- JRun? [40]
- WebLogicServer? [40]
- Oracle? [20]
- Tuxedo? [50]
- MQ Series? [50]
- CICS (ECI/EPI)? [20]
- LDAP? [50]

Können folgende PROTOKOLLE gemessen werden?

- HTTP? [50]
- HTTPS? [50]
- T3? [40]
- T3S? [40]
- LDAP? [20]
- Tuxedo? [50]
- MQ Series? [50]
- CICS (ECI/EPI)? [20]
- SQL? [20]
- RMI-IIOP? [30]
- IP? [40]
- CORBA? [30]
- COM/DCOM? [30]
- JDBC? [30]

Hardware

- Liegt der Speicherverbrauch pro virtuellem Benutzer laut Herstellerangaben unter 1 MB? [30]
- Liegt der gemessene Speicherverbrauch pro virtuellem Benutzer unter 1 MB? [30]

Netzwerkmonitoring

- Kann Netzwerkmonitoring über den eigenen Client durchgeführt werden? [30]
- Gibt es vom Hersteller des zu messenden Servers eine Schnittstelle für Netzwerkmonitoring zum Lasttesttool? [30]
- Greift das Lasttesttool auf Standard Netzwerkmonitoring Schnittstellen zurück? [30]

Recording-Verfahren

- Zum Aufzeichnen wird die API des Clients (z.B. Browser) verwendet? [40]
- Aufgezeichnet wird auf Netzwerkkartenebene? [20]
- Es wird Proxy-Recording verwendet? [30]

A.1.2 Bedienbarkeit

Installation / Administration

- Ist die Installation Assistenten- oder Scriptgesteuert? [40]
- Ist die Installation einfach und ohne Nachbearbeitung durchführbar? [30]
- Ist das Produkt ohne Windows Servicepack oder Patches lauffähig? [20]
- Ist die Deinstallation einfach? [20]
- Löscht die Deinstallation die Applikation restlos? [10]
- Ist die Installation Assistenten- oder Scriptgesteuert? [40]
- Ist die Installation einfach und es ist keine Nacharbeit erforderlich? [30]
- Ist die Applikation ohne Kernel- oder andere Patches lauffähig? [20]
- Ist die Deinstallation der Applikation einfach? [20]
- Werden bei der Deinstallation der Applikation alle Komponenten gelöscht? [10]
- Werden bei der Installation alle wichtigen Konfigurationsparameter erfasst? [30]
- Werden bei der Installation alle wichtigen Konfigurationsparameter erfasst? [30]
- Ist der Administrationsaufwand während der Verwendung gering? [20]
- Ist der Administrationsaufwand während der Verwendung gering? [20]

Bedienung allgemein

- Ist die Bedienung intuitiv? [30]
- Gibt es einen integrierten Workflog (Assistent oder Ähnliches) ? [20]
- Ist die Einarbeitungsphase in die Werkzeuge (z.B. Toolbars) kurz? [40]
- Gibt es eine hilfreiche und ausführliche Dokumentation? [40]
- Gibt es ein einführendes Tutorium? [30]
- Ist die Onlinehilfe ausführlich und hilft sie weiter? [30]
- Sind Abstürze der oder von der Software verursachte selten? [30]
- Ziehen Abstürze KEINE schlimmere Folgen nach sich? [50]

A.1.3 Funktionalität

Testskripterstellung

- Ist das Aufzeichnen eines Scriptes einfach? [40]
- Ist das Skript nachvollziehbar? [40]
- Wurde eine bekannte Programmiersprache verwendet oder ähnelt die verwendete einer solchen? [30]
- Sind in der verwendeten Skriptsprache logische Funktionsblöcke definierbar? [40]
- Sind in der verwendeten Skriptsprache Kommentare einfügbar? [30]
- Sind mit der verwendeten Skriptsprache Synchronisationspunkte definierbar? [30]
- Ist das erzeugte HTTP-Skript browserunabhängig? [20]
- Sind in einer Recording-Session mehrere Skript erstellbar? [20]
- Sind die Skripte speicherbar? [40]
- Werden die Testskripte übersichtlich abgelegt? [30]
- Definiert das Skript Round-Trip Messpunkte? [30]
- Ist ein T3-Protokoll-Recording ohne Zusatzprodukt vorhanden? [30]
- Sind selbstgeschriebene Testclients benutzbar? [40]
- Falls ja, sind Java-Clients nutzbar? [30]
- Falls ja, sind Ansi C/C++ Clients nutzbar? [30]
- Falls ja, sind Visual Basic-Clients nutzbar? [20]
- Kommt der Testclients dabei ohne ein Wrapping aus? [30]
- Können aufgezeichnete Eingabewerte parametrisiert werden? [40]
- Falls ja, sind Zufallswerte möglich? [40]
- Falls ja, sind Datentabellen definierbar? [50]
- Falls Datentabellen definierbar sind, können diese mit einem Wizard erzeugt werden? [30]
- Falls Datentabellen definierbar sind, sind diese mit einem Texteditor erzeugbar? [30]
- Falls Datentabellen definierbar sind, sind diese aus Excel importierbar? [30]
- Falls Datentabellen definierbar sind, sind diese aus einer ODBC-Datenbank mit Hilfe eines SQL-Statements importierbar? [20]
- Sind die Skripte nach dem Beenden per erneutem "Rekording" erweiterbar? [20]
- Sind die Skripte nach dem Beenden manuell erweiterbar? [40]
- Können nachträglich Kommentare in die Skripte eingefügt werden? [30]

- Können in den Skripten später logische Funktionsblöcke definiert werden? [40]
- Sind in den Skripten später Synchronisationspunkte definierbar? [40]
- Können bestehende Skripte getrennt und zusammengeführt werden? [10]

Testmanagement

- Ist eine Anbindung an ein Testmanagementtool vorhanden? [20]
- Ist eine Anbindung an ein Versionierungstool vorhanden? [20]
- Ist eine Anbindung an eine Fehlerdatenbank vorhanden? [30]

Testablauf

- Wird das Lasttestszenario zentral gesteuert? [20]
- Ist der Lasttest verteilbar? [30]
- Berücksichtigt die Verteilung die Agent-Ressourcen? [20]
- Sind mehrere Benutzergruppen definierbar? [50]
- Können Benutzergruppen unterschiedliche Skriptszenarien ausführen? [40]
- Kann eine Benutzergruppe verschiedene Skripte ausführen? [40]
- Sind mehrere Benutzergruppen auf unterschiedlichen Agenten (Clients) möglich? [40]
- Ist eine Benutzergruppe auf unterschiedliche Agents verteilbar? [40]
- Ist ein statischer Lasttest möglich (alle Benutzer beginnen gleichzeitig)? [50]
- Ist ein Ramp-Up Lasttest möglich (die Benutzeranzahl steigt stufenweise)? [50]
- Ist die Benutzeranzahl funktionsbasiert (Eine Funktion definiert zu jedem Zeitpunkt die Anzahl der zu simulierenden Benutzer)? [40]
- Das Abspielen des Szenarios ermöglicht die Simulation beliebiger Browser? [30]
- Java Exceptions werden als Fehler aufgefangen? [20]
- Javascript Exceptions werden als Fehler aufgefangen? [20]
- Können unterschiedliche Netzbandbreiten simuliert werden? [40]
- Kann IP-Spoofing simuliert werden? [40]
- Ist ein Lasttest ins Internet möglich? [40]
- Kann der Lasttest nach einem Absturz fortgeführt werden? [30]
- Lassen sich die Tests hierarchisch (in Baumform) gliedern? [30]
- Lassen sich die Messdaten in einem Graphen oder einer Tabelle online beobachten? [40]
- Kann ein Messdatum einfach der Messung hinzugefügt werden? [30]

- Sind SNMP Objekte grafisch ermittelbar? [40]
- Wird ein Logging der Messdaten in eine CSV Datei geschrieben? [30]
- Wird ein Logging der Messdaten in eine ODBC Datenbank geschrieben? [20]
- Kann das Skript nach Aufzeichnung und Anpassung einem Testlauf unterzogen werden, um zu prüfen, ob es fehlerfrei ist? [20]
- Ist ein Skriptdebugging möglich? [20]

Testauswertung

- Werden alle Messwerte in einem Graphen angezeigt? [50]
- Sind die Messkurven frei skalierbar? [30]
- Ist eine statistische Auswertung der Messung möglich? [40]
- Können die Auswertungsdaten exportiert werden? [40]
- Falls die Daten exportiert werden Können, können die Auswertungsdaten in Excel exportiert werden? [40]
- Falls die Daten exportiert werden Können, können die Auswertungsdaten in HTML exportiert werden? [30]
- Falls die Daten exportiert werden Können, können die Auswertungsdaten in CSV exportiert werden? [20]
- Werden die Ergebnisse versioniert? [20]
- Können verschiedene Testläufe verglichen werden? [40]

Messdaten

- Können generelle Kenndaten vom Controller-Rechner ermittelt werden (RAM, CPU-Auslastung, etc...)? [20]
- Können generelle Kenndaten vom Agenten-Rechner ermittelt werden (RAM, CPU-Auslastung, etc...)? [20]
- Können generelle Kenndaten von Windows gemessen werden? [50]
- Können generelle Kenndaten von Solaris gemessen werden? [40]
- Können generelle Kenndaten von HP UX gemessen werden? [20]
- Können generelle Kenndaten von Linux gemessen werden? [40]
- Können generelle Kenndaten von Tandem gemessen werden? [20]
- Können generelle Kenndaten von IBM S/390 gemessen werden? [20]

A.1.4 Sonstiges

Support

- Wird persönlicher Support geboten? [40]
- Wird telefonischer Support geboten? [30]
- Falls ja, ist die Erreichbarkeit gut? [30]
- Wird E-Mail Support geboten? [30]
- Falls ja, ist die Antwortzeit entsprechend kurz? [30]
- Ist die Qualität der Antworten zufriedenstellend? [20]
- Besitzt das Supportpersonal genügend technisches Know How? [20]

Referenzen

- Wird das Produkt auf dem Markt häufig eingesetzt? [30]
- Wird das Produkt durch Drittanbieter unterstützt? [50]

A.2 Evaluierungsergebnisse

Kriterium	Mercury Loadrunner 6.5	Apache JMeter 1.9.1	Testmaker 4.0.6	Dieseltest 1.0.21	OpenSTA 1.4.2.34	Siege 2.5.9
TECHNIK						
TESTUMFELD						
Javatest	✓	✓	✓	✓	✓	✓
HTML-Test	✓	✓	✓	✓	✓	✓
Übertrag Subkategorie (100)	100	100	100	100	100	100
	100%	100%	100%	100%	100%	100%
BETRIEBSSYSTEME						
Windows	✓	✓	✓	✓	✓	✗
Solaris	Ⓢ	✓	✓	✗	✗	✓
Linux	Ⓢ	✓	✓	✗	✗	✓
HP-UX	Ⓢ	✓	✓	✗	+	✓
IBM S/390	✗	✓	✓	✗	+	✗
Tandem	✗	✓	✓	✗	+	✗
Übertrag Subkategorie (190)	110	190	190	50	50	120
	57.9%	100%	100%	26.3%	26.3%	63.2%
PRODUKTE						
Apache	✓	✗	✗	✓	✗	✗
JRun	✗	✗	✗	✗	✗	✗
WebLogic	Ⓢ	✗	✗	✗	✗	✗
Oracle	✓	✗	✗	✗	✗	✗
Tuxedo	Ⓢ	✗	✗	✗	✗	✗
MQ Series	✓	✗	✗	✗	✗	✗
CICS (ECI/EPI)	✗	✗	✗	✗	✗	✗
LDAP	✗	✗	✗	✗	✗	✗
Übertrag Subkategorie (310)	155	0	0	40	0	0
	50%	0%	0%	12.9%	0%	0%
PROTOKOLLE						
HTTP-Protokoll	✓	✓	✓	✓	✓	✓
HTTPS-Protokoll	✓	✓	✓	✓	✓	✓
T3-Protokoll	✓	✗	✗	✗	✗	✗
T3S-Protokoll	✓	✗	✗	✗	✗	✗
LDAP-Protokoll	✓	✓	✗	✗	✗	✗
Tuxedo-Protokoll	✓	✗	✗	✗	✗	✗
MQ Series-Protokoll	✓	✗	✗	✗	✗	✗
CICS-Protokoll	✗	✗	✗	✗	✗	✗
SQL-Protokoll	✗	✗	✗	✗	✗	✗
RMI-Protokoll	Ⓢ	✗	✗	✗	✗	✗
IP-Protokoll	✓	✗	✗	✗	✗	✗
CORBA-Protokoll	✓	✗	✗	✗	✗	✗
COM/DCOM-Protokoll	✓	✗	✗	✗	✗	✗
JDBC-Protokoll	✓	✓	✗	✗	✗	✗
Übertrag Subkategorie (500)	445	150	100	100	100	100
	89%	30%	20%	20%	20%	20%

Kriterium	Mercury Loadrunner 6.5	Apache JMeter 1.9.1	Testmaker 4.0.6	Dieseltest 1.0.21	OpenSTA 1.4.2.34	Siege 2.5.9
HARDWARE						
Mem/User (Hersteller)	✗	+	+	+	+	+
Mem/User (Messung)	✓	✗	+	✓	✓	✓
Übertrag Subkategorie (60)	30	0	0	30	30	30
	50%	0%	0%	50%	50%	50%
NETZWERKMONITORING						
Client NW-Monitoring	✗	✗	✗	✗	✓	✗
Server NW-Monitoring	Ⓢ	✗	✗	✗	✗	✗
API NW-Monitoring	✓	✗	✗	✗	✗	✗
Übertrag Subkategorie (90)	45	0	0	0	30	0
	50%	0%	0%	0%	33.3%	0%
RECORDINGVERFAHREN						
Recording Client API	✓	✗	✓	✓	✓	✗
Recording Netzwerkkarte	✗	✗	✗	✗	✗	✗
Recording Proxy	✓	✓	✗	✗	✗	✗
Übertrag Subkategorie (90)	70	30	40	40	40	0
	77.8%	33.3%	44.4%	44.4%	44.4%	0%
Übertrag Kategorie (1340)	955	470	430	360	350	350
	71.3%	35.1%	32.1%	26.9%	26.1%	26.1%

BEDIENBARKEIT

INSTALLATION/ADMINISTRATION

Win-Setup	✓	✗	✓	✓	✓	+
Win Nacharbeit nach Setup	✓	✓	✓	✓	✓	+
Läuft ohne SP?	✗	✓	✓	✓	✓	+
Win simples Deinstall	✓	✓	✓	✓	✓	+
Win Deinstall komplett	✓	✓	✓	✓	✓	+
Unix Installsript	+	✗	✓	+	+	✗
Unix Install ohne Nacharbeit	+	✓	✓	+	+	✗
Unix ohne Patch	+	✓	✓	+	+	✓
Unix Deinstall einfach	+	✓	✓	+	+	✗
Unix Loesch komplett	+	✓	✓	+	+	✗
Win Parametererfassung	✓	✓	Ⓢ	✓	✓	+
Unix Parametererfassung	+	✓	Ⓢ	+	+	✗
Win Adminaufwand	✓	✓	✓	✓	✓	+
Unix Admin gering	+	✓	✗	+	+	✓
Übertrag Subkategorie (340)	150	260	290	170	170	40
	44.1%	76.5%	85.3%	50%	50%	11.8%

BEDIENUNG ALLGEMEIN

Intuitive Bedienung	✓	✓	Ⓢ	✓	✓	✗
Integrierter Workflow	✗	✗	✗	✗	Ⓢ	✗
Schnelle Einarbeitung	✓	✓	✓	✓	✓	✓
Übertrag Subkategorie (90)	70	70	55	70	80	40
	77.8%	77.8%	61.1%	77.8%	88.9%	44.4%

Kriterium	Mercury Loadrunner 6.5	Apache JMeter 1.9.1	Testmaker 4.0.6	Dieseltest 1.0.21	OpenSTA 1.4.2.34	Siege 2.5.9
DOKUMENTATION						
Dokumentation	✓	✓	Ⓢ	Ⓢ	✓	✓
Tutorium	✗	✓	Ⓢ	✗	✓	✗
Onlinehilfe	Ⓢ	✓	Ⓢ	Ⓢ	Ⓢ	Ⓢ
Übertrag Subkategorie (100)	55	100	50	35	85	55
	55%	100%	50%	35%	85%	55%
STABILITÄT						
Seltene Abstürze	✓	✓	✓	✓	✓	✓
Abstürze nicht schlimm	✓	✓	✓	✓	✓	✓
Übertrag Subkategorie (80)	80	80	80	80	80	80
	100%	100%	100%	100%	100%	100%
Übertrag Kategorie (610)	355	510	475	355	415	215
	58.2%	83.6%	77.9%	58.2%	68%	35.2%

FUNKTIONALITÄT

TESTSKRIPTERSTELLUNG

Einfaches Rekording	✓	Ⓢ	✓	✓	✓	✗
Script nachvollziehbar	✓	✓	✓	✓	✓	✓
Bekannte Skriptsprache	✓	✓	✓	✗	✓	✓
Funktionsblöcke definierbar	✓	✓	✓	✗	✓	✗
Kommentare einfügbar	✓	✓	✓	✗	✓	✓
Sync-Punkte definierbar	✓	✗	✗	✗	✓	✗
HTTP-Skript browserunabhängig	✓	+	✓	✗	+	✓
Mehrere Skripte/Session	✗	✗	✗	✗	✗	✗
Skripte speicherbar	✓	✓	✓	Ⓢ	✓	✗
Übersichtliche Skriptablage	✓	Ⓢ	✗	Ⓢ	✓	✗
Roundtrip-Messpunkte	✓	Ⓢ	✗	✗	✓	✗
T3-Recording	✓	✗	✗	+	+	✗
Eigene Testclients	✓	✓	✓	✗	✗	✗
Java-Clients	✓	✓	✓	+	✗	✗
C/C++ Clients	✓	✓	✓	+	✗	✗
VB-Clients	✗	✓	✓	+	✗	✗
Kein Client-Wrapping	✓	+	✗	+	+	+
Parametrisierung	✗	✓	Ⓢ	✗	✓	Ⓢ
Zufallswert-Parameter	Ⓢ	✓	Ⓢ	+	✓	✗
Datentabellen	✓	✓	✗	+	✓	✗
Datentabellen/Wizard	✓	✗	+	+	✓	+
Datentabellen/Text	✓	✓	+	+	✓	+
Datentabellen/Excel	✗	✗	+	+	✗	+
Datentabellen/ODBC	✗	✗	✗	+	✓	+
Skript per Rekording erweiterbar	✓	✓	✗	✗	✗	✗
Skript manuell erweiterbar	✓	✓	✓	✓	✓	✓
Skript kommentierbar	✓	✓	✓	✗	✓	✓
Logische Funktionsblöcke	✓	✓	✓	✗	✓	✗
Sync-Punkte	✓	✗	✗	✗	✓	✗
Skripttrennung/-zusammenführung	Ⓢ	Ⓢ	Ⓢ	✗	✓	✗
Übertrag Subkategorie (950)	795	645	515	155	680	210
	83.7%	67.9%	54.2%	16.3%	71.6%	22.1%

Kriterium	Mercury Loadrunner 6.5	Apache JMeter 1.9.1	Testmaker 4.0.6	Dieseltest 1.0.21	OpenSTA 1.4.2.34	Siege 2.5.9
TESTMANAGEMENT						
Testmanagement Anbindung	✗	✗	✗	✗	✓	✗
Versionierung Anbindung	Ⓢ	✗	✗	✗	✗	✗
Fehlerdatenbank Anbindung	Ⓢ	✗	✗	✗	✗	✗
Übertrag Subkategorie (70)	25	0	0	0	20	0
	35.7%	0%	0%	0%	28.6%	0%
TESTABLAUF						
Szenario zentral gesteuert	✓	✓	+	✓	✓	✗
Lasttest verteilbar	✓	✓	+	✗	✓	✗
Auf Agent-Ressourcen warten	✗	✗	+	✗	✗	+
Mehrere Benutzergruppen	✓	✓	+	✗	✓	✗
Unterschiedliche Szenarien	✓	✓	+	✗	✓	✗
Verschiedene Skripte	✓	✓	+	✗	✗	✗
Mehrere Gruppen/Agent	✓	✓	+	✗	✓	✗
Gruppe auf Agents verteilbar	✓	✗	+	✗	✓	✗
statischer Lasttest	✓	✓	+	✓	✓	✓
Ramp-Up Lasttest	✓	✓	+	✓	✓	✗
Benutzeranzahl funktionsbasier	✗	✗	+	✗	✗	✗
Simulation beliebiger Browser	✓	✗	+	✗	✗	✗
Java Exceptions als Fehler	✗	+	+	✗	✗	✗
JS Exceptions als Fehler	✗	+	+	✗	✗	✗
Bandbreitensimulation	Ⓢ	✗	+	✗	✗	✗
IP-Spoofing	✓	✗	+	✗	✗	✗
Intrenet-Lasttest	✓	✓	+	✓	✓	✓
Testfortführung nach Absturz	✓	✓	+	✗	✗	✗
Hierarchische Tests	✗	✓	+	✗	✗	✗
Online-Beobachtung	✗	✓	+	✗	✓	✗
Messdatum einfügen	✓	✓	+	✗	✓	✗
SNMP Objekte grafisch	✓	✗	+	✗	✗	✗
Messdaten-Log in CSV	✗	✗	+	✗	✗	✗
Messdaten-Log in ODBC	✗	✗	+	✗	✗	✗
Skript-Testlauf	✓	✗	✓	✗	✓	✓
Skriptdebugging	✓	✗	✓	✗	✓	✗
Übertrag Subkategorie (870)	630	490	40	160	470	110
	72.4%	56.3%	4.6%	18.4%	54%	12.6%
TESTAUSWERTUNG						
Alle Messwerte in Graphen	✓	Ⓢ	+	Ⓢ	✓	✗
Messkurven skalierbar	✓	✗	+	Ⓢ	✓	+
Statistische Auswertung	✓	Ⓢ	+	Ⓢ	✓	Ⓢ
Export der Ergebnisse	✓	Ⓢ	+	Ⓢ	✓	✗
Ergebnisexport in Excel	✓	Ⓢ	+	✗	✓	+
Ergebnisexport in HTML	✓	✗	+	✗	✗	+
Ergebnisexport in CSV	✓	✗	+	✗	✗	+
Ergebnisversionierung	✗	✗	+	✗	✓	✗
Vergleichbarkeit der Tests	✓	Ⓢ	+	Ⓢ	Ⓢ	✗
Übertrag Subkategorie (310)	290	105	0	100	240	20
	93.5%	33.9%	0%	32.3%	77.4%	6.5%





Kriterium	Mercury Loadrunner 6.5	Apache JMeter 1.9.1	Testmaker 4.0.6	Dieseltest 1.0.21	OpenSTA 1.4.2.34	Siege 2.5.9
MESSDATEN						
Kenndaten Controllerrechner	✓	✗	+	✗	✗	✗
Kenndaten Agentenrechner	✓	✗	+	✗	✗	✗
Kenndaten Windows	✓	✗	+	✗	✗	✗
Kenndaten Solaris	✓	✗	+	+	+	✗
Kenndaten HP UX	✓	✗	+	+	+	✗
Kenndaten Linux	✗	✗	+	+	+	✗
Kenndaten Tandem	✗	✗	+	+	+	✗
Kenndaten BM S/390	✗	✗	+	+	+	✗
Übertrag Subkategorie (230)	150 65.2%	0 0%	0 0%	0 0%	0 0%	0 0%
Übertrag Kategorie (2430)	1890 77.8%	1240 51%	555 22.8%	415 17.1%	1410 58%	340 14%

SONSTIGES

SUPPORT

Support persönlich	✓	✗	✓	✗	Ⓢ	✗
Support telefonisch	✓	✗	✗	✗	✗	✗
Erreichbarkeit Telefon	Ⓢ	✗	+	+	+	+
Support E-Mail	✓	✓	✓	✗	✓	✓
Antwortzeit E-Mail	Ⓢ	Ⓢ	Ⓢ	+	Ⓢ	✓
Qualität der Antworten	✓	Ⓢ	Ⓢ	+	Ⓢ	✓
Know How Supportpersonal	✓	✓	Ⓢ	+	Ⓢ	+
Breiter Produkteinsatz	✓	✗	Ⓢ	✗	Ⓢ	✗
Unterstützung durch Drittanbie	✓	✗	Ⓢ	✗	Ⓢ	✗
Übertrag Subkategorie (280)	250 89.3%	75 26.8%	145 51.8%	0 0%	125 44.6%	80 28.6%
Übertrag Kategorie (280)	250 89.3%	75 26.8%	145 51.8%	0 0%	125 44.6%	80 28.6%
ENDERGEBNIS (4660)	3450 74%	2295 49.2%	1605 34.4%	1130 24.2%	2300 49.4%	985 21.1%

LEGENDE

 Ja
  Eingeschränkt
  Nein
  Irrelevant

Anhang B

Performance-Studie

B.1 Aufbau der Testskripte

Testskript 1 (Professorenskript)

- Download des Grundlagenmodulkatalogs
- Anmelden als „prof/prof“
- Forum ▷ Dozentenbereich ▷ Postformular anzeigen ▷ Neuen Thread erzeugen
- 5 Klicks
- Abmelden

Testskript 2 (Studenten)

- Login als „sutdent/student“ fehlerhaft eingeben ▷ zurück ▷ Korrektes Login
- Professoren ▷ Druckansicht ▷ zurück
- Forum ▷ Web Performance ▷ Test A1 ▷ Post Formular anzeigen ▷ Posting ▷ Vorschau ▷ eintragen
- Webmail ▷ Fehlermeldung übergehen ▷ Account anlegen¹ ▷ hinzufügen ▷ klick auf neuen Account
- 5 Klicks
- Abmelden

¹Name: Test / PopServer: pop.onlinehome.de / SMTPServer: auth.mail.onlinehome.de / User: cc34428738 / PW: coin

Abbildungsverzeichnis

1.1	Webbasiertes Evaluierungstool	5
1.2	Webbasierte Evaluationsauswertung	5
2.1	Apache JMeter - Übersicht	10
2.2	Apache JMeter - Graphische Auswertung	11
2.3	Testmaker nach erfolgreichem Abspielen eines aufgezeichneten Skriptes . . .	13
2.4	OpenSTA Script Modeler beim Abspielen eines aufgezeichneten Skriptes . .	15
2.5	Lasttestvorbereitung im OpenSTA Commader	16
2.6	Ergebnisauswertung mit OpenSTA	17
2.7	Die minimalistisch anmutende Benutzeroberfläche von Dieseltest	19
2.8	Skriptdarstellung alá Dieseltest	19
2.9	Testergebnisse und Auswertung mit Dieseltest	20
3.1	Lasttesttools - Technik	25
3.2	Lasttesttools - Bedienbarkeit	26
3.3	Lasttesttools - Funktionalität	27
3.4	Lasttesttools - Sonstiges	27
3.5	Lasttesttools - Gesamtvergleich	28
3.6	Lasttesttools - Verteilung	28
4.1	Schmatischer Aufbau der Testumgebung	31
5.1	Startseite unseres Testportals	32
6.1	Loadrunner - Transaktionsvergleich	35
6.2	Loadrunner - Durchsatz	36
6.3	Loadrunner - Übersicht	37
6.4	JMeter - Tabellenauswertung	38
6.5	OpenSTA - Response-Times	39
6.6	Auswertung der Siege Tests	41
6.7	Siege - Systemauslastung bei 25 Benutzern	41

List of Algorithms

1	Apache JMeter Testskriptausschnitt: Timer-Element	10
2	Siege Testskript Professoren	23
3	Loadrunner Foreneintragsfunktion	34