Stefan Huth

(E-Mail: Stefan.Huth@prodyna.de)
arbeitet bei der PRODYNA AG als Software Engineer und Quality Analyst. Neben der Entwicklung und Qualitätssicherung von Software beschäftigt er sich mit Themen wie Cloud Computing.

Funktionale Webtests mit Selenium in der Cloud

Funktionale Tests helfen dabei Nutzerverhalten zu simulieren und Webanwendungen aus dem Browser heraus zu testen. Dabei wird aber oftmals keine Last auf die Webanwendung erzeugt, weil die dafür genutzten Frameworks viele Ressourcen benötigen. Der Artikel zeigt eine Möglichkeit, wie man mit Hilfe von Cloud Computing funktionale Tests durchführen und mehrere parallele Zugriffe auf die Webanwendung simulieren kann um Fehler wie Memory Leaks, Verhalten unter Last oder Deadlocks zu ermitteln.

Funktionale Tests als automatisierten Nutzerersatz

Für die Entwicklung von Webanwendungen sind automatisierte Tests unersetzlich geworden. Dabei gibt es unterschiedliche Arten und Ansätze von Tests. In den meisten Fällen beginnt die Qualitätssicherung mit den Tests des Source Codes. Für eine durchgehende Sicherstellung der Qualität folgen weitere Tests durch Frameworks wie HtmlUnit oder HTTPUnit, bei dem die gesamte Applikation mit Client und Server getestet werden kann. Um aber ein möglichst aussagekräftiges Szenario für das Testen zu erreichen, welches einen realen Nutzer simuliert, also auch Tests der JavaScript- und Ajax-Komponenten, müssen funktionale Tests der Webseiten her. Ein normales Vorgehen für funktionale Tests ist das Bereitstellen der Webseite in einem produktionsnahen System. Das Framework für die funktionalen Tests greift per HTTP auf diese Webseite zu und führt vordefinierte Tests aus, welche die Anwendungsfälle eines späteren produktiven Einsatzes darstellen. Abbildung 1 zeigt exemplarisch die Durchführung eines solchen Tests mit Selenium Tests, Selenium Remote und Selenium Server. Die Selenium Tests senden die Befehle mittels Selenium Remote an den Selenium Server. Dieser startet intern einen Browser, zum Beispiel Firefox oder Internet Explorer, ruft die Webseite auf und führt die Tests aus.

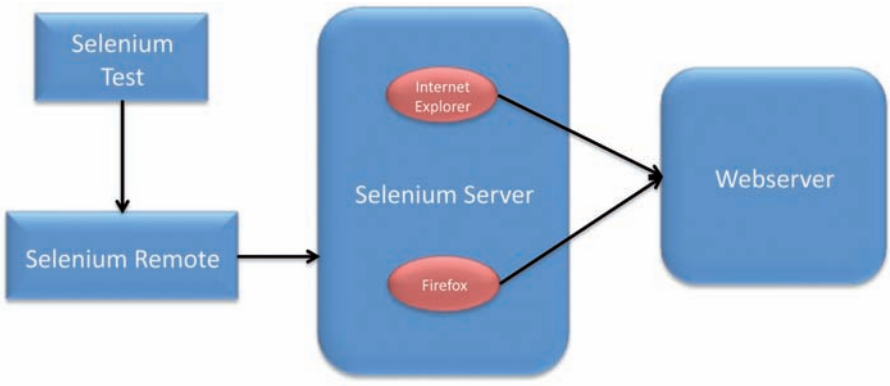

Abbildung 1: Aufbau eines funktionalen Tests mit Selenium

Die so durchgeführten Tests haben den Vorteil, dass ein Nutzerverhalten gut abgebildet werden kann. Es werden Webseiten im Browser geladen und die Elemente der Webseite so angesprochen, wie es auch ein Nutzer in seinem eigenen Browser machen würde. Der Nachteil ist allerdings, dass auf diese Weise keine Last auf die Anwendung und den Webserver bzw. Servletcontainer kommt. Eine Lösung für mehr Last wäre der Einsatz mehrerer Rechner oder Selenium-Server auf einem Rechner, die parallele Zugriffe auf die Webanwendung erzeugt.

Selenium Grid als Hilfsmittel für mehrere Parallele Tests

Eine Lösung für das Verwenden mehrerer Selenium Server ist der Einsatz von Selenium Grid. Hierbei wird ein Hauptknoten, der Selenium Hub, verwendet, welcher alle Tests per Selenium Remote entgegennimmt. Im Gegensatz zu der Variante mit mehreren Selenium Server ist damit ein zentraler Punkt definiert, der alle funktionalen Tests entgegennimmt. Für das Durchführen der Tests werden einzelne Knoten gestartet. Diese Knoten nehmen die Tests von dem Hub entgegen und führen diese wie ein einzelner Selenium Server aus. Der Hub verteilt dabei die Tests auf die jeweiligen

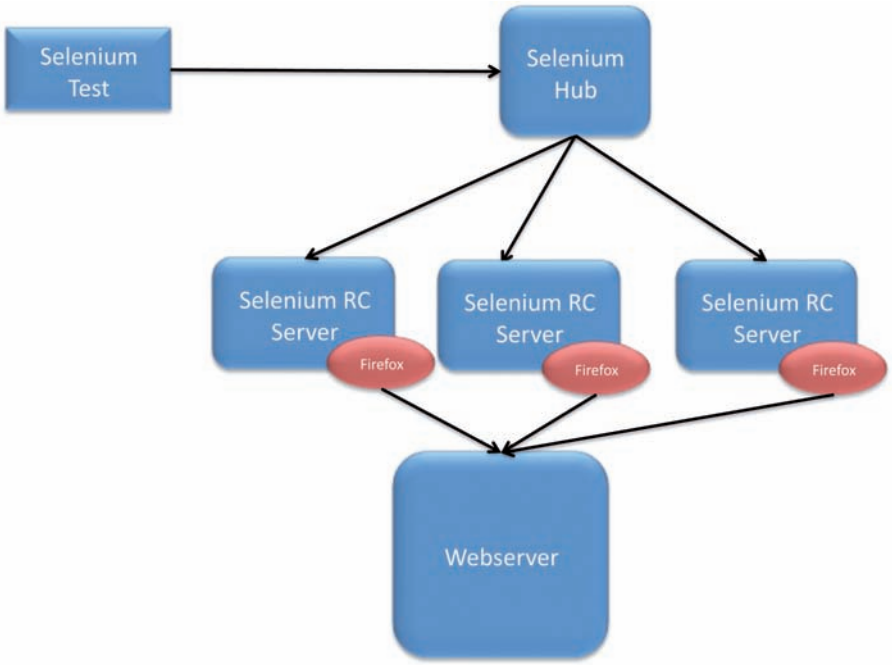

Abbildung 2: Selenium Grid

Knoten. Dabei erkennt der Hub, welche Knoten bereits ausgelastet sind. Sind alle Knoten ausgelastet, werden die ausstehenden Tests in eine Warteschlange gesetzt und werden ausgeführt, sobald ein Knoten frei wird. Weitere Unterscheidungen werden vorgenommen, je nachdem ob der Knoten mit Firefox oder Internet Explorer per Konfiguration verknüpft wurde. Da jeder Knoten über einen Port und eine URL angemeldet und angesprochen wird, können mehrere parallele Tests gestartet werden und auf einem Rechner ausgeführt werden. Abbildung 2 zeigt den Aufbau einer Testumgebung mit Selenium Grid. Allerdings kommen auch moderne Rechner mit mehreren Knoten und aktiv laufenden Tests schnell an ihre Grenzen. Nutzt man mehrere Rechner innerhalb eines Netzwerks, können diese einzelne Knoten enthalten und sich an den zentralen Hub anmelden um Tests auszuführen. Diese Konstellation kann zwar mehrere parallele Tests ausführen, aber ist in einem lokalen Netzwerk beheimatet, wodurch die Anbindung des Servers nicht berücksichtigt werden kann.

Die Cloud als Tester

Die Variante mit dem Rechner im lokalen Netzwerk ist eine kleine Steigerung der Ressourcen für das Testen der Webanwendung. Allerdings hat man nicht immer die Masse an Ressourcen zur Verfügung, die notwendig ist, um eine größere Last zu erzeugen. Eine Möglichkeit um kurzzeitig mehr Rechenleistung zur Verfügung zu stellen, ist die Nutzung der Private Cloud. Hierbei werden vorgefertigte Images mit Betriebssystem und Knoten für das Selenium Grid bereitgestellt und gestartet. Damit können zur Verfügung stehende Server für Tests genutzt werden, und nach dem Test wieder für andere Aufgaben genutzt werden. Die Anschaffung und der Betrieb der Hardware ist meistens recht teuer. Werden die Server nicht ständig ausgelastet, entstehen hohe Kosten, welche oftmals den Benefit, welcher durch Lasttests entsteht, nicht ausgleichen

Diese Kosten übersteigen meistens die Mittel, die viele Firmen oder QA Abteilungen zur Verfügung haben. Die Budgets werden meist mit der Entwicklung der Software und der Durchführung der Tests ausgelastet.

Die Lösung dafür sind Public Clouds von großen Anbietern wie Amazon, Google oder IBM. Diese Variante bietet ausreichend Rechenleistung, kostet aber nur so lange Geld, wie sie auch tatsächlich genutzt wird. Einen weiteren Vorteil bieten Public Clouds durch ihre Anbindung. Da die Server nicht innerhalb des eigenen Netzwerks stehen, wird ein weiterer Aspekt der produktionsnahen Tests erfüllt. Neben Spezialanwendungen für das Testen in der Cloud bieten die größeren Anbieter in der Regel keine vorgefertigten Images für Selenium Grid an.

Anbieter wie Amazon haben aber den Vorteil, dass man sich spezielle Images zusammenstellen kann, welche gespeichert werden und, falls notwendig, zeitnah gestartet werden können. Die QA Abteilung hat somit die Möglichkeit ein Image

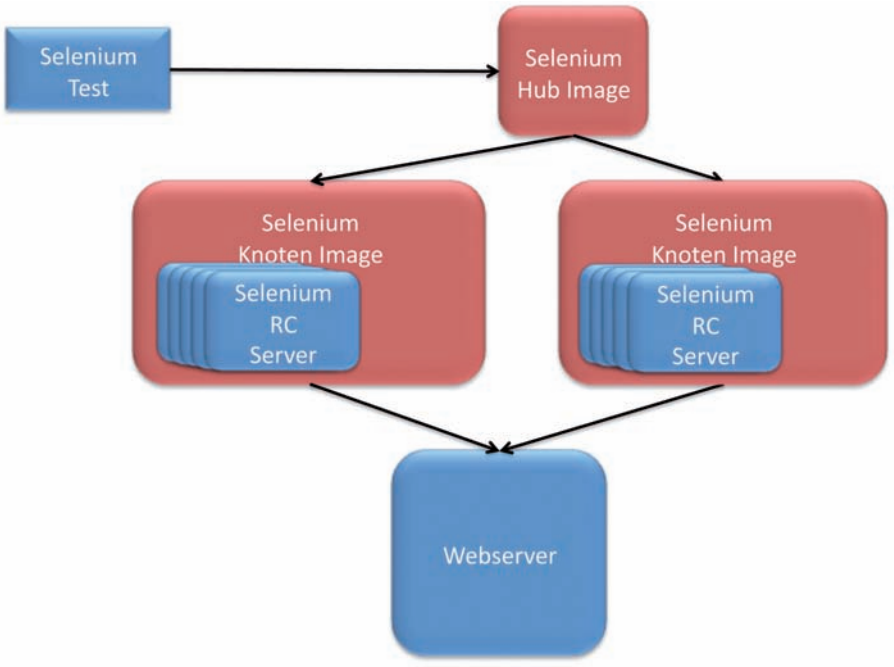

Abbildung 3: Selenium Grid mit Amazon EC2

in Form des Selenium Hubs und ein Image für die Knoten zu erstellen.

Wenn die Selenium Tests geschrieben sind, wird der Selenium Hub hochgefahren und die Tests können auf die Instanz verschoben werden. Je nach Anforderungen werden mehrere Instanzen für die Knoten gestartet. Auf diesen Instanzen kann nun eine beliebige Anzahl an Knoten an den Hub angemeldet werden. Je nach Leistungsfähigkeit, die bei Amazon EC2 gewählt werden kann, können mehrere Knoten auf einer Instanz gestartet werden. Sind mehr Knoten notwendig, wird eine weitere Instanz für die Knoten hochgefahren. So erreicht man eine Konfiguration von Selenium Grid, welche eine Vielzahl, beispielsweise 50, Selenium Knoten enthält. Jetzt kann der Selenium Test ausgeführt werden. Das Load Balancing von Selenium Grid verteilt die Anfragen auf die jeweiligen Knoten. Sind die bestehenden Knoten nicht ausreichend, kann man leicht mehr Instanzen starten, und auf diese Weise die Ressourcen dynamisch skalieren. Auf diese Weise kann eine Webapplikation samt JavaScript und Ajax-Komponenten mit mehreren parallelen Zugriffen getestet werden. Natürlich lässt sich dadurch nicht die gleiche Last erzeugen wie mit reinen Performance-Tests, welche direkt auf das HTTP Protokoll zurückgreifen. Die Variante mit Selenium erzeugt aber einen Test, welcher näher an der Realität liegt und alle beteiligten Komponenten, auch den Webserver, testet.

Cloud Computing als Erweiterung der eigenen Hardware.

Auch wenn diese Variante des Testens mit Selenium und Amazon EC2 nur eine Variante des Testens mit Cloud Computing darstellt, zeigt dies doch die Möglichkeiten, welche Public Clouds auch der Qualitätssicherung eröffnen. Skalierbare Ressourcen, welche nur einen Bruchteil der Kosten verursachen im Vergleich zu lokaler Hardware, bieten auch Projekten mit weniger Budget die Möglichkeit produktionsnahe Lasttests durchzuführen. Man erreicht nicht unbedingt die Auslastung, welche mit anderen Frameworks, wie JMeter, möglich ist, aber man erhält einen realistischeren Eindruck, wie sich die zu testende Webapplikation unter der Last realer Nutzer verhält. Auf diese Weise kann das gesamte System mit Anwendungsfällen geprüft werden und unangenehme Überraschungen bei einem Live Gang vermieden werden. ■

Quellen

[1] Selenium – http://seleniumhq.org/
[2] Selenium Grid – how it works - http://selenium-grid.seleniumhq.org/how_it_works.html
[3] Amazon EC2 - http://aws.amazon.com/ec2/