Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projekt

Florian Burka

Seminar Ringvorlesung - Skalierung von Webapplikationen durch Agenten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Florian Burka

Seminar Ringvorlesung - Skalierung von Webapplikationen durch Agenten

Seminarausarbeitung im Rahmen des Master Projekts
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Prüfer:
Prof. Dr. Ing. Birgit Wendholt und Prof. Dr. rer. nat. Stephan Pareigis
Betreuender Professor: Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 21. Juni 2009

Inhaltsverzeichnis

1 Einführung

Agenten sind seit langem ein Thema in Veröffentlichungen. Mit *The Agent Network Architecture (ANA)* (Maes [1991]) war schon 1991 eine erste Architektur für Agenten und Netzwerke von Pattie Maes veröffentlicht worden. Seither wurden Agenten und Netzwerke immer wieder in verschiedenen Zusammenhängen betrachtet (siehe hierzu auch Burka [2008]).
Es gibt auch wie in Luck u. a. [2005] beschriebene Ideen von virtuellen Gesellschaften von Agenten welche die Steuerung abgeschlossener Aufgabengebiete übernehmen.[1] Mein Wunsch ist es eine solche virtuelle Agentenwelt in kleinem Rahmen zu schaffen.

Als Aufgabengebiet soll hierfür das Skalieren von Anwendungen, speziell von Webanwendungen herangezogen werden.
Die Skalierung von Webanwendungen selbst ist ein gut durchdrungenes Thema und wird in Büchern wie *The Art of Capacity Planning* (Allspaw [2008]) ausführlich behandelt.

Mit dem Aufkommen der von Services wie Amazon EC2[2] entstand durch die Virtualisierung von Hardware die Möglichkeit Skalierung rein virtuellen Entitäten zu überlassen.

Nun kann die Skalierung einer Agentengesellschaft überlassen werden.

Die Idee zu dieser Arbeit entstand durch mein Interesse an Agenten und Enterprise Computing sowie durch Bücher wie *The Art of Capacity Planning* (Allspaw [2008]) und Romane wie Accelerando (Stross [2005]). Auch Veröffentlichungen wie Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing) (Luck u. a. [2005]) zeigen, dass die Agententechnologie noch einige Veränderungen in der Softwarelandschaft nach sich ziehen wird. Ob in Reinform oder aber als Spender von Ideen und Konzepten wie in der geplanten Masterarbeit wird sich zeigen.

[1] In der Roadmap ist ein sehr schönes Beispiel eines potentiell guten Aufgabengebietes: Die Koordination von Zulieferern eines Automobilherstellers. Die Preisverhandlungen und Liefertermine werden zwischen den Unternehmen in einer virtuellen Agentengesellschaft ausgehandelt und sind so wesentlich dynamischer gestaltbar.

[2] `http://aws.amazon.com/ec2/`

2 Projektidee

Um die Aufgabe der Skalierung von Webanwendungen nicht dem Zufall zu überlassen oder auch Effekten wie dem *Slashdotting*[3] nicht ausgeliefert zu sein soll ein Framework entwickelt werden welches skalierbare Webanwendungen mithilfe von Agententechnologie skaliert.

Hierzu wird eine virtuelle Agentengesellschaft (Luck u. a. [2005]) geschaffen welche sich dieser Aufgabe annimmt. Die Agenten agieren sozusagen im Sinne ihrer Erzeuger als Advocate Agents[4] und kümmern sich um die Verteilung der Anwendung sowie der Beschaffung von Ressourcen. Dazu nutzen sie die Möglichkeiten der Cloud[5].

Die Ideen von Agenten sollen in dieser Arbeit allerdings eher als Architekturgrundlage herangezogen werden. Es soll nicht Ziel sein die konkrete Implementation Anhand von Agentenframeworks zu vollziehen sondern die Metaphern zu nutzen.

2.1 Zielsetzung

Ziel der Masterarbeit ist es ein einfaches Framework für die automatisierte Verteilung und Skalierung von Webanwendungen zu entwickeln und dieses anhand einer Beispielanwendung zu evaluieren. Für die grundlegenden Architekturentscheidungen sollen viele Ideen aus der agentenorientierten Softwareentwicklung übernommen und vereinfacht implementiert werden.

Das Ergebnis der Arbeit soll ein Framework für weitestgehend selbstskalierende (Web-) Anwendungen sein.

[3] `http://en.wikipedia.org/wiki/Slashdotted`
[4] Wie in Burka [2009] beschrieben.
[5] `http://de.wikipedia.org/wiki/Cloud_Computing`

3 Realisierung

Für die erforderliche Anwendung soll auf bewährte und mit bekannte Technologien zurückgegriffen werden. So wird ein einfacher Webshop in Java mit JSF, Hibernate und JMS entwickelt werden, welche in einem Tomcat auf einem JBoss unter Linux laufen.

Die Virtualisierung soll auf einer vorhandenen Lösung aufbauen. Diese Lösung soll von Agenten integriert werden.

Die Agenten übernehmen die Aufgabe eines Systemadministrators der sich um die Beobachtung des laufenden Betriebs kümmert und neue Server bei Bedarf hinzufügt oder entfernt.

Das Framework soll am Ende verschiedene Lastszenarien durchlaufen um zu zeigen ob dieser Ansatz ein gangbarer Weg ist.

3.1 Webanwendung

Es wird ein einfacher Webshop programmiert, welcher verteilt werden kann. Dieser Webshop soll dabei sowohl horizontal als auch vertikal gut skalierbar sein. Für die vertikale Skalierbarkeit sind zunächst folgende Schichten geplant:

- Webserver
- Business
- Backend / Persistenz

Die Webserver kümmern sich hierbei lediglich um die Anzeige und die Verwaltung von Web-Sessions.
Die Businessschicht im Frontend ist für die Abwicklung von Bestellungen verantwortlich. Sie stellt die Bestellungen zusammen und kann gegebenenfalls Verfügbarkeiten und Zahlungsmittel im Backend prüfen.
Das Backend verwaltet Lagerbestände, wickelt Bestellungen ab, oder ist an Bezahlsysteme angebunden.

Die Verteilung der Anfragen soll ein einfacher Load Balancer übernehmen (siehe auch 5.2).

3.2 Transaktionen

Ein spannender Bereich innerhalb dieser Arbeit sind verteilte Transaktionen. Da die Dienste auf unterschiedlichsten Servern an unterschiedlichsten Orten laufen können, muss ein Mechanismus entwickelt werden, der die entstehenden Daten konsistent hält. Dies zu untersuchen könnte aber eine weitere Masterarbeit füllen, daher soll dies nur am Rande betrachtet werden.

Um diese Problematiken zu umschiffen soll in der zu entwickelnden Anwendung eine Shared-Nothing-Architektur (Stonebraker [1986]) verwendet werden und dedizierte Server für einzelne Händler oder einzelner Lager eingesetzt werden. Die Kommunikation soll mittels eines Messagingsystems transparent gehalten werden. Hierfür ist es geplant JBoss Messaging[6] einzusetzen.

3.3 Virtualisierung

Die Virtualiserung der Gastsysteme soll unter Linux erfolgen.
Zunächst müssen hierfür Lösungen für die Verwaltung von Virtualisierungsumgebungen evaluiert werden. Untersucht werden sollen zunächst die frei verfügbaren Lösungen Convirture[7], OpenNebula[8] und Ganeti[9]. Hierbei wird insbesondere ein Augenmerk auf eine einfache Integration in das Framework gelegt. Des Weiteren sollte die Anwendung neben Grundeigenschaften wie Stabilität und Sicherheit auch eine einfache Verteilung der Gastsysteme zur Verfügung stellen. OpenNebula scheint derzeit der geeignetste Kandidat zu sein, da er von Haus aus eine XML-RPC Api mitbringt sowie eine Unterstützung von Xen[10], KVM[11] und Amazon EC2[12].

Zunächst sollen hierbei die Blades in der HAW-Hamburg als Testbett genutzt werden (siehe auch 5.2).

[6] http://www.jboss.org/jbossmessaging/
[7] http://www.convirture.com/
[8] http://www.opennebula.org
[9] http://code.google.com/p/ganeti/
[10] http://www.xen.org/
[11] http://www.linux-kvm.org
[12] http://aws.amazon.com/ec2/

3.4 Agenten

Die Agenten sollen als Metapher genommen werden um die Verteilung der Anwendung aus Architektursicht zu beschreiben.

Agenten können Dienste benötigen, bereitstellen oder sie wissen wo Dienste aufzufinden sind. Diese Informationen werden auf einem oder mehreren virtuellen Marktplätzen ausgetauscht. Damit folgt die Grundidee dem Gedanken nahezu sämtlicher verteilter Systeme wie dem WWW, Corba oder der Idee von Webservices und einer UDDI Registry[13].

Es wird zunächst eine einfache Version des Konzepts implementiert. Hierfür werden folgende Agenten programmiert:

- Agenten die Serverkapazitäten anbieten.
- Agenten die eine Anwendung betreuen. Sie nutzen die Serverkapazitäten.

Diese Agenten unterhalten sich auf einem virtuellen Marktplatz. Auf diesem können Angebote für Serverkapazitäten eingestellt werden.

3.4.1 Agenten als Anwendungsbetreuer

Die Agenten überwachen ihre jeweiligen System. Zunächst werden auf Seiten des Betriebssystems die Festplattennutzung (IO-Wait), die CPU-Auslastung (load), die Arbeitsspeicherauslastung sowie die Netzwerkauslastung gemessen. Auf Seiten der Anwendung werden zusätzliche die Anzahl der Transaktionen (angezeigte Webseiten, Businesstransaktionen, Bezahlvorgänge, Speicheranfragen) sowie deren Ausführungszeiten gemessen.

Durch die Beobachtung dieser Werte und (zunächst voreingestellter) Maxima für die Antwortzeiten können die Agenten neue Serverkapazitäten anfordern und die Anwendung skalieren.

Sollte ein Agent kaum noch Anforderungen bearbeiten, kann er auch Serverkapazitäten wieder freigeben und seine Aufgaben einem anderen Agenten übergeben.

[13] Wobei die Verwendung von einer UDDI Registry als Yellow Pages für diesen Anwendungsfall noch genauer evaluiert werden muss. Es würde aber, wenn es nicht viel Verwaltungsaufwand mit sich bringt, ein weiteres Problem lösen.

3.4.2 Agenten die Serverkapazitäten bereitstellen

Als Server werden zunächst nur die Blades aus der HAW Hamburg genommen, es ist jedoch angedacht dass verschiedene Agenten Server mit unterschiedlichen Leistungsmerkmalen anbieten.
So unterscheiden sich Server in CPU-Leistung, Arbeitsspeicher, Netzwerkanbindung, Festplattendurchsatz und Speicherkapazität. Als weiterer, in der Masterarbeit aber nicht weiter betrachteter Faktor, kommt die Zuverlässigkeit und die Vertrauenswürdigkeit eines Servers.

4 Auswertung

In der Masterarbeit soll untersucht werden ob dieser Architekturvorschlag in Form des zu programmierenden Frameworks seinen Anforderungen gerecht wird und die entwickelte Webanwendung entsprechend den Anforderungen skaliert.

Hierfür sollen verschiedene Szenarien durchgespielt werden und die Leistungsdaten der Anwendung genauer betrachtet werden.

Zunächst sind folgende zwei Szenarien gedacht:

- Langsam wechselnde Lasten die einen normalen Wochenverlauf darstellen sollen mit Hochzeiten tagsüber und nahezu keinem Betrieb des Nachts.
- Plötzlich wechselnde Lasten.

Für diese Szenarien soll die automatische Skalierung beobachtet werden und das Framework dahingehend optimiert werden, dass die Anfragen mit möglichst wenig Ressourcen aber auch ohne große Wartezeiten beantwortet werden.

5 Ausblick auf die Masterarbeit

5.1 Risiken

Da das Thema sehr umfangreich ist werde ich mich auf eine einfache Implementierung in einem mir gut bekannten Umfeld beschränken. Als Java J2EE Applikation welche in virtuellen Maschinen unter Linux läuft. Die virtuellen Maschinen sollen zunächst auf den Blades in der HAW laufen.

Die infrastrukturellen Teile wie die Auswahl der Virtualisierungslösung und die Entwicklung einer verteilt skalierenden Webanwendung sollten nicht zu viel Zeit in Anspruch nehmen um genug Zeit für die interessanteren Themen und deren Auswertung zu lassen.

5.2 Erweiterungsmöglichkeiten

Das Framework kann um die Verteilung der Aufgaben sowohl innerhalb der Architektur als auch der von außen kommenden Anfragen erweitert werden. Die Lastverteilung durch die Agenten lässt diese die Auslastung der Anwendung noch feiner steuern.

Die Verteilung der Ressourcen kann durch eine Marktwirtschaft innerhalb der Agentengesellschaft modelliert werden. In diesem Beispiel könnten die Agenten durch die Vermittlung von Bestellungen Geld verdienen, welches sie für den Betrieb ihrer Anwendungen benötigen[14]. Agenten müssen dann, wenn sie neue Server für ihre Anwendung starten, dem diese Anwendung betreuenden Agenten entsprechend Geld mitgeben. Sollte ein Server nicht mehr benötigt werden kann der Server heruntergefahren werden und dem Agenten der die Aufgabe übernimmt das restliche Geld zugeteilt werden.

Die Nachrichtenverteilung kann auch direkt durch Agenten oder mit Agenten als Boten geschehen (siehe auch Burka [2008]).

[14] In der Hoffnung, dass die Agenten, anders als viele Menschen, auch trotz angespartem Geld noch bereit sind sich und ihr Geld zu teilen falls die derzeitige Last dies erfordert ;).

5.3 Fazit

Die angestrebten Ziele sind nur ein Anfang.

Eine Anbindung von weiteren ServiceProvidern in Form von Amazon EC2 oder anderen Cloud Service Providern und der Test des Frameworks in hinsicht auf eine Weltweite Verteilung einer oder mehrerer Anwendungen wäre ein interessanter nächster Schritt.

Die Auswahl von recht trägen Technologien wie J2EE nehme ich bewusst in Kauf: hier bringe ich viel Erfahrung mit und kann die damit gewonnene Zeit nutzen um den Fokus auf andere Details zu legen.

Vielleicht hat die die Masterarbeit ja meinen Wunschtitel:

Agents in the Cloud

Literaturverzeichnis

Allspaw 2008
ALLSPAW, John: *The Art of Capacity Planning: Scaling Web Resources*. O'Reilly Media, Inc., 2008 http://portal.acm.org/citation.cfm?id=1457542. – ISBN 0596518579, 9780596518578

Burka 2008
BURKA, Florian: *Agenten in Netzwerken*. HAW-Hamburg, 2008

Burka 2009
BURKA, Florian: *Advocate Agents*. HAW-Hamburg, 2009

Luck u. a. 2005
LUCK, M. ; MCBURNEY, P. ; SHEHORY, O. ; WILLMOTT, S.: *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink, 2005

Maes 1991
MAES, Pattie: The Agent Network Architecture (ANA). In: *SIGART*, 1991

Stonebraker 1986
STONEBRAKER, Michael: The case for shared nothing. In: *Database Engineering* 9 (1986), 4–9. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.5370

Stross 2005
STROSS, Charles: *Accelerando*. Ace Hardcover, 2005 http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0441012841. – ISBN 0441012841

Alle Verweise auf Quellen im Internet wurden am 25.02.2009 auf ihre Aktualität überprüft.