Laufzeitunterstützung für mobilen Code

Dr. Holger Peine

Fraunhofer-Institut Experimentelles Software Engineering
peine@iese.fraunhofer.de
Sauerwiesen 6, 67661 Kaiserslautern

Abstract: Mobiler Code ist ein Modell der Rechnerkommunikation, bei dem nicht durch den Austausch von Nachrichten interagiert wird, sondern der gesamte Interaktionscode zum Partner-Rechner geschickt und dort lokal ausgeführt wird. Das verspricht dynamischere und weniger vom Netzwerk abhängige Anwendungen. Mobile Agenten sind Prozesse aus mobilem Code, die sich selbstständig durch das Netzwerk bewegen. In dieser Arbeit wurden die abstrakten Eigenschaften, die Anwendungsmöglichkeiten und die benötigte Laufzeitunterstützung für mobilen Code und mobile Agenten untersucht und ein solches Laufzeitsystem, genannt Ara, entwickelt. Ara zeichnet sich dadurch aus, dass verschiedene Programmiersprachen zur Anwendungsentwicklung an einen gemeinsamen, sprachunabhängigen und effizienten Systemkern angeschlossen werden können, und dass die Agenten jederzeit unter voller Erhaltung ihres internen Zustands ihren Rechner wechseln und weiterlaufen können. Eine Anwendung von Ara zur Suche in einem verteilten Datenbestand wird beschrieben, bei der mobile Agenten messbare Geschwindigkeitsvorteile gegenüber herkömmlichen Methoden erzielen. Allerdings scheint es nur wenige Anwendungen zu geben, die den spezifischen Mehraufwand für mobile Agenten gegenüber mobilem Code rechtfertigen, weshalb diese Arbeit vor allem für mobilen Code, weniger aber für mobile Agenten interessante Zukunftsaussichten sieht.

1. Mobiler Code und mobile Agenten

In heutigen Rechnernetzen führt die Zunahme von Geräten und Anwendungen immer wieder zu Engpässen bei der Bandbreite. Da ein stark steigender Anteil von künftigen Netzverbindungen drahtlos mit vergleichsweise geringer Bandbreite sein wird, wird Bandbreite auf absehbare Zeit knapp bleiben. Neben der Bandbreite liegt die zweite Hauptanforderung an Rechnernetze in immer flexibleren und dynamischen Diensten. Vernetzte Geräte unterscheiden sich immer stärker in ihren Kapazitäten hinsichtlich Kommunikation, Rechenkraft, Energievorrat und Benutzerschnittstellen, und ebenso unterschiedlich sind die Funktionen, die ihre Benutzer von diesen Geräten erwarten. Mobile Geräte, die ständig lokale Netzwerk- und Dienste-Strukturen betreten und verlassen, erhöhen diese Komplexität noch weiter. Die Netze müssen ihre Dienste dynamisch anpassen und erweitern, möglichst ohne menschliche Eingriffe, und natürlich mit so wenig Bandbreite wie möglich.

Konventionelle Rechnerkommunikation, die auf dem Remote-Procedure-Call Modell (RPC) beruht, ist für diese Anforderungen nicht ideal, da RPC eine beständige Netzverbindung über die Dauer der Interaktion hinweg erfordert, spürbare Bandbreite, und ein festes, gemeinsames Protokoll auf beiden Seiten. Alle diese drei Anforderungen passen schlecht zu der zuvor skizzierten bandbreitenschwachen und dynamischen Netzumgebung. *Mobiler Code* ist ein alternatives Modell, das auf der Übertragung des Interaktionscodes des einen Partners zum Rechner des anderen beruht, wo er dann abläuft, ohne noch eine Netzverbindung zu benötigen, potentiell Bandbreite spart, und nicht mehr als ein allgemeines Laufzeitsystem für mobilen Code voraussetzt. *Mobile Agenten* sind die allgemeinste Form von mobilem Code: Ein autonomer Prozess, der nacheinander auf verschiedenen Rechner ablaufen kann, und sich selbstständig mit seinem gesamten Code und Zustand durch ein heterogenes Netz bewegt ("Migration"), wie in Abb. 1 dargestellt.

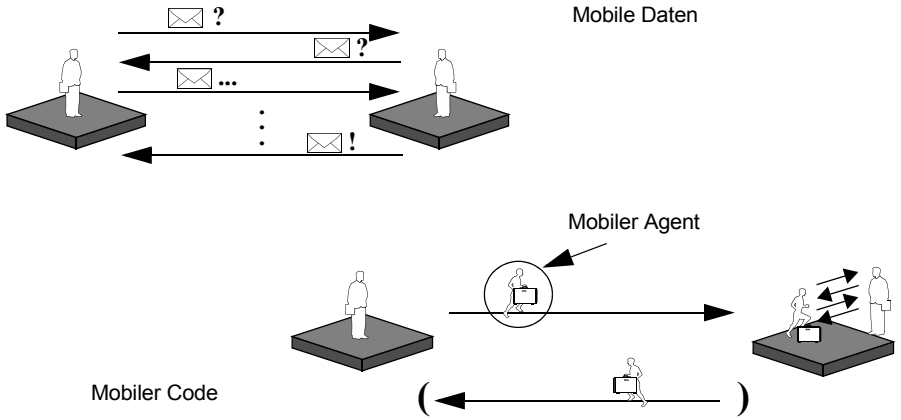

Abb. 1. Mobile Daten und mobiler Code

In dieser Arbeit wurden fünf konstituierende Eigenschaften mobiler Agenten herausgearbeitet, die sie von verwandten Konzepten wie Prozessmigration, mobilen Objekten, RPC oder verteilten Betriebssystemen unterscheiden: Die *Allgemeinheit* ihres Codes, die *Asynchronität* ihrer Ausführung, ihre *willkürliche Mobilität* und ihr *Ortsbewusstein*, die *Heterogenität* der unterliegenden Rechner- und Netzplattformen, sowie das Überschreiten von Grenzen zwischen verschiedenen *Schutzbereichen*. Der Konferenzband mit [PS97] und die Bände der Folgejahre geben einen Überblick über die Forschung zu mobilen Agenten.

2. Mobile-Agenten-Systeme

Fremden Code auf einem Rechner auszuführen erfordert einigen Aufwand. Mobile Agenten stellen neuartige Anforderungen an Laufzeitsysteme, an erster Stelle die Sicherheit und Portabilität der Ausführung des Agenten-Codes. Hierfür sind spezielle, oft komplexe Laufzeitsysteme erforderlich, *Mobile-Agenten-Systeme* genannt (MAS). Ein solches MAS muss auf jedem Rechner vorhanden sein, der mobile Agenten beherbergen soll.

Die Portabilität des Agenten-Codes ist in der Praxis unverzichtbar, da es unrealistisch wäre, verschiedene Versionen eines Agenten für jede Plattform vorzuhalten, die ihm in einem großen Netz wie dem Internet begegnen könnte. Deshalb muss der Agent in einer portablen Repräsentation verschickt werden, die dann das lokale MAS ausführt, indem es eine Art Interpretierung, Kompilierung, oder eine Kombination von beidem benutzt. Neben dem ausführbaren Code gelten dieselben Portabilitätsanforderungen auch für die Repräsentation der Daten und des Zustands des Agenten.

Die zweite offensichtliche Anforderung bei der Aufnahme eines mobilen Agenten auf einem Wirtssystem ist die Sicherheit. Das MAS muss die Ausführung des Gast-Codes genau überwachen, um beabsichtigte oder unbeabsichtigte Schäden, Ausspähen, Überlastung und all die anderen wohlbekannten Netzwerk-Angriffe zu verhindern. Umgekehrt kann auch der Agent schutzbedürftig gegenüber dem Wirtssystem sein; für dieses noch schwierigere Problem sind überhaupt nur Teillösungen bekannt.

Zusätzlich zu den zentralen Anforderungen Portabilität und Sicherheit muss ein MAS noch eine Reihe weiterer Funktionen bieten, um praxistauglich zu sein, so etwa Kommunikation der Agenten untereinander, Kontrolle und Verwaltung der Agenten. Zahlreiche andere aus Verteilten Systemen bekannte unterstützende Dienste können ebenfalls nützlich sein, z.B. Fehlertoleranz, Gruppenkommunikation, Verzeichnisdienste, Schlüsselverwaltung usw.

3. Das Ara-System

Ein solches MAS, das im Zuge dieser Arbeit entwickelt wurde, ist das *Ara*-System (“Agents for Remote Action”) [Pei02b, Pei03]. Abgesehen von den oben beschriebenen notwendigen Funktionen, zeichnet sich Ara gegenüber anderen MAS dadurch aus, dass verschiedene verbreitete und stark unterschiedliche Programmiersprachen zur Agentenprogrammierung unterstützt werden [Pei97a], dass Agenten “stark” (d.h. ohne Beeinflussung ihres Kontrollflusses) migrieren können [Pei97b], und dass besonderen Wert auf eine effiziente Implementierung gelegt wurde. Ara steht zum freien Download einschließlich allen Source-Codes und vielfältiger Dokumentation zur Verfügung [Pei].

Das Programmiermodell von Ara besteht aus Agenten, die sich zwischen sogenannten Places bewegen oder dort aufhalten und bestimmte Dienste nutzen, die vom gastgebenden Rechner oder lokalen Agenten angeboten werden. Agenten können jederzeit während ihrer Ausführung den Place wechseln und dann ihre Ausführung an derselben Stelle in ihrem Programm fortsetzen. Davon abgesehen, werden Agenten meistens wie gewöhnliche Programme implementiert, d.h. sie arbeiten mit dem Dateisystem, der Netzwerk- und der Benutzerschnittstelle. Die Ara-Systemarchitektur verzichtet bewusst auf höher abstrahierte agentenspezifische Konzepte wie etwa Unterstützung für intelligente Interaktionsmuster, die der Anwendung überlassen werden, oder komplexe verteilte Dienste wie man sie in verteilten Betriebssystemen findet, da solche Funktionen mit der Mobilität und Autonomie von mobilen Agenten im Konflikt stehen.

3.1 Systemkern und Interpreter

Die meisten MAS nutzen den gleiche Grundansatz für Portabilität und Sicherheit: Agenten laufen auf einer virtuellen Maschine ab, meistens ein Interpreter mit einem Laufzeitsystem, was sowohl die Details der Wirtsplattform verbirgt, als auch die Aktionen der Agenten auf eine eingeschränkte Umgebung begrenzt. Auch Ara folgt diesem Ansatz.

Konkret werden Ara-Agenten in einer interpretierten Programmiersprache formuliert und in einem Interpreter für diese Sprache ausgeführt, der von einem speziellen Laufzeitsystem für mobile Agenten unterstützt wird, in Ara *Systemkern* genannt. Die Trennung von Kern und Interpretern ist charakteristisch für Ara: Die sprachabhängige Funktionalität, wie etwa die Extraktion des C-spezifischen Zustands eines in der Sprache C programmierten Agenten, wird im Interpreter isoliert, während die sprachunabhängige Funktionalität, wie etwa die Extraktion des allgemeinen Zustands eines Ara-Agenten und seine anschließende Migration, im Systemkern isoliert wird. Zur besseren Kompatibilität mit bestehenden Programmiermodellen und bestehender Software schreibt Ara keine bestimmte Programmiersprache vor, sondern bietet stattdessen eine Schnittstelle, um vorhandene Sprachen anzubinden. Eine saubere Aufgabentrennung ermöglicht es, mehrerer Interpreter für verschiedene Programmiersprachen nebeneinander auf dem gemeinsamen Systemkern zu betreiben, der seine Funktionen Agenten aller Sprachen in gleicher Weise zur Verfügung stellt. Abgesehen von den Migrationsfunktionen, zählen die Verwaltung der Agenten, ihre Kommunikation und Persistenz sowie verschiedene Sicherheitsfunktionen zu den Aufgaben des Systemkerns. Abb. 2 zeigt eine Konfiguration aus Kern, Agenten und Interpretern für Sprachen namens *A* und *B*.

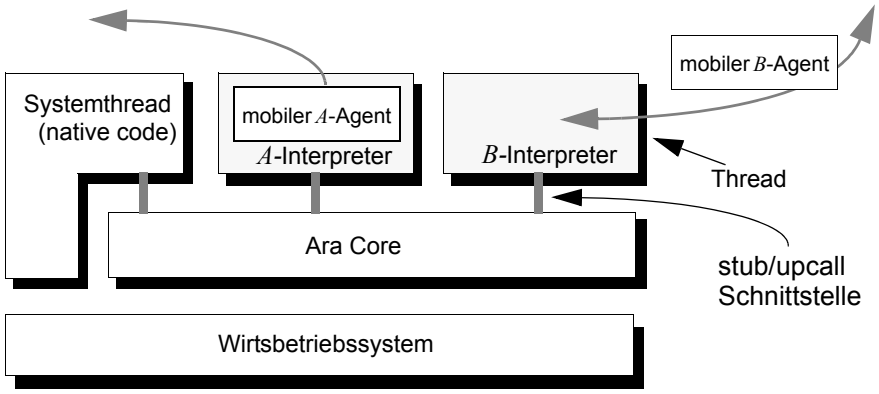

Abb. 2. Grobansicht der Ara-Systemarchitektur

Die Funktionalität des Systemkerns ist auf ein unverzichtbares Minimum beschränkt, indem Funktionen höherer Abstraktionsstufe von speziellen Server-Agenten angeboten werden. Das komplette Ensemble von Agenten, Interpretern und Systemkern läuft als ein einziger Anwendungsprozess auf einem Standard-Betriebssystem.

Um die Praktikabilität und Flexibilität dieses Ansatzes zu demonstrieren, wurden Interpreter für drei recht verschiedene Programmiersprachen angebunden, die ein breites Spektrum von Anwendungen abdecken: Die Skriptsprache Tcl, C/C++ (durch Vorkompilierung in ein interpretierbares Bytecode-Format [Sto95]), und Java.

Agenten werden als separater Kontrollfluss (Thread) ausgeführt, solange sie sich auf einem Rechner befinden, und werden transparent in eine portable Form umgewandelt, sobald sie eine Migration beginnen. Das System setzt weitere Threads für interne Aufgaben ein („Systemthreads“) ein[1], um die Architektur modularer zu gestalten. Der Einsatz von Threads im Gegensatz zu Betriebssystemprozessen hält die Verwaltung der Agenten vollständig unter Kontrolle des Systemkerns, steigert die Performance, und erleichtert die Portierung von Ara auf andere Betriebssysteme.

Um einen gegebenen Interpreter für eine gewisse Programmiersprache *A* an den Ara-Kern anzupassen, müssen Aufrufschnittstellen (Stubs) für die Programmierschnittstelle des Kerns in *A* implementiert werden, sowie — im umgekehrter Richtung — einige Funktionen zur Verwaltung des Interpreters durch den Kern (Upcalls). Die Stubs führen hauptsächlich Formatkonvertierungen und ähnliche schnittstellentypische Anpassungen durch (z.B. auch Adressumsetzungen). Unter den Interpreter-Upcalls sind die zentralen Funktionen diejenigen zur Extraktion und Restauration des Zustands eines ablaufenden Interpreters, wie es für die Migration eines interpretierten Agenten nötig ist. Schließlich muss der Interpreter noch den Kern bei der präemptiven Ausführung der Agenten-Programme unterstützen, indem er regelmäßig eine Kern-Funktion zur Zeitscheibenüberwachung aufruft. Diese Konstruktion ermöglicht es, dass die Agenten präemptiv ausgeführt werden und ihre Ausführungskontexte trotzdem immer maschinenunabhängig sind.

3.2 Kommunikation

Da einer der Hauptgründe für die Einführung mobiler Agenten in der Vermeidung entfernter Kommunikation lag, betont Ara besonders die lokale Kommunikation. Zwei einfache und effiziente Mechanismen stehen dafür zur Verfügung, synchrone Nachrichten und ein Tupel-Speicher. Zur Nachrichtenkommunikation bietet der Kern das Konzept der *Service Points*. Das sind Rendezvous-Punkte mit symbolischen Namen, an denen Agenten als Clients und Server durch den Austausch von synchronen n:1 Auftrags- und Ergebnis-Nachrichten interagieren können. Jeder Auftrag ist mit dem Namen des Clients markiert, und der Server-Agent kann dies bei der Entscheidung über die Antwort benutzen.

Der zweite Kommunikationsmechanismus zwischen Ara-Agenten ist der *Tupel-Speicher*, ein persistenter Speicher für (Schlüssel, Wert)-Paare, der zur asynchronen Kommunikation genutzt werden kann, was sich als besonders geeignet für die asynchrone Struktur vieler Mobile-Agenten-Anwendungen erwiesen hat. Der Tupel-Speicher wird durch einen System-Thread implementiert, der als Server-Agent an einem allgemein bekannten

1. Falls solche Threads vertrauenswürdig und nicht mobil sind, können sie auch in nativen Maschinencode kompiliert werden, um maximale Performance zu erreichen. Native-Code-Threads werden genau wie ihre interpretierten Geschwister als Agenten behandelt, abgesehen davon, dass sie (normalerweise) nicht mobil sind.

Service Point zugänglich ist. Agenten können dort mit einem Zugriffsschlüssel markierte Daten hinterlegen, die später von anderen Agenten im Besitz des passenden Schlüssels gelesen werden können.

3.3 Sicherheit

Die Sicherheit von Ara beruht auf der Authentisierung von Agenten-Benutzern, -Herstellern und Rechnern. Hierauf aufbauend, bietet ein flexibles System von sogenannten *Places* mit lokaler Autorisierungspolitik eine feinkörnige, aber konzeptuell einfache Autorisierung für mobile Agenten, die einen Rechner besuchen [Pei98]. Agenten können über authentisierte und verschlüsselte Verbindungen übertragen werden.

Das zentrale Autorisierungskonzept in Ara ist die *Allowance*. Eine Allowance ist eine Liste von Zugriffsrechten auf verschiedene Systemressourcen, wie etwa Dateien, CPU-Zeit, Hauptspeicher, oder Plattenplatz. Die Elemente einer solchen Liste geben „Zugriffsgrenzen“ an, die quantitativ (z.B. für CPU-Zeit) oder qualitativ (z.B. für erlaubte Netzwerk-Domains) sein können. Wenn ein Agent zu einem Place migrieren möchte, gibt er seine für den Aufenthalt dort erwünschte Allowance an, und der Place entscheidet, welche Allowance er dem Bewerber tatsächlich zugestehen möchte. Diese Entscheidung wird in Form einer beliebigen, anwendungsdefinierten Funktion für jeden Place formuliert, die die lokale Allowance auf der Grundlage der Identitätsangaben und der Wünsche des Bewerbers berechnet.

Das Sicherheitskonzept von Ara schützt also gegen die Gefahren von Abhören, Hochstapelei, lokalem übermäßigen Ressourcenverbrauch (denial-of-service) und allgemein unerlaubten Zugriffen durch Agenten, und gewährleistet die Sicherheit des Rechners auf flexible, bequeme, und effiziente Weise.

4. Eine Anwendung zur verteilten Suche

Zahlreiche Anwendungen wurden für mobile Agenten vorgeschlagen. Es fehlte allerdings eine systematische Analyse der abstrakten Eigenschaften einer Anwendung, die notwendig sind, um tatsächlich einen nicht nur zufälligen Vorteil durch mobile Agenten zu erzielen. Die Analyse in dieser Arbeit hat ergeben, dass eine vorteilhafte Anwendungssituation für mobile Agenten sowohl eine *schwache Netzanbindung* als auch einen Bedarf für *unvorhersehbare Funktionalität* aufweisen sollte.

Als hauptsächliche Klassen von solchen Anwendungen wurden in dieser Arbeit das entfernte Filtern von Daten, Mobilcomputeranwendungen und die dynamische Rekonfigurierung entfernter Rechner herausgearbeitet. Für eine dieser Klassen, das entfernte Filtern, wurde eine in Umfang und Nutzwert realistische Anwendung zur verteilten Recherche im Usenet News Netzwerk mit Ara-Agenten entwickelt. Dabei wurden erhebliche Leistungsgewinne durch den Einsatz mobiler Agenten zum Filtern entfernter Datenbestände im Vergleich zu einer konventionellen Implementierung gemessen, die auf RPC und lokalem Filtern beruhte [Pei02a].

Usenet News ist ein weltweit verteiltes Netz von Servern, die untereinander von menschlichen Benutzern für die Allgemeinheit geschriebene Artikel zu vielfältigen Themen austauschen. Die Benutzer greifen auf Usenet zu, indem sie sich mit Hilfe von Client-Anwendungen mit einem lokalen Server verbinden und die dort vorhandenen Artikel lesen. Falls sie einen neuen Artikel schreiben, wird der ebenfalls ausgehend von diesem Server ins Usenet injiziert. Usenet Server tauschen regelmäßig Artikel untereinander aus und löschen auch lokal gespeicherte Artikel, beides nach ihrer eigenen lokalen Strategie. Im Ergebnis besitzt jeder Server eine andere und ständig wechselnde Teilmenge der gesamten Usenet-Daten. Usenet ist somit eine Art von global verteilter Datenbank mit vielen Redundanzen und einer schlechten Schnittstelle für entfernte Anfragen, und damit ein lohnendes Ziel für das Suchen mit mobilen Agenten.

Die Grundidee der Suchanwendung, die in (interpretiertem) C++ implementiert ist, liegt darin, einen mobilen Agenten zu einem entfernten Usenet Server zu schicken, um dort lokal dessen Datenbestand nach „interessanten“ Artikeln zu durchsuchen. Dieser Ansatz kann Netzbandbreite und -latenz einsparen im Vergleich zu einer nicht-mobilen Lösung, wo alle Artikel erst in Gänze zur einer stationären Suchanwendung übertragen werden müssen, die dann die Filterung an einem zentralen Ort durchführt. Darüberhinaus kann ein mobiler Suchagent auch eine bessere Datenabdeckung erreichen, indem er mehrere Server besucht wie in Abb. 3 angedeutet, wobei er etwas „Intelligenz“ zur Ermittlung von Servern einsetzen kann, die potentiell weitere interessante Artikel enthalten.

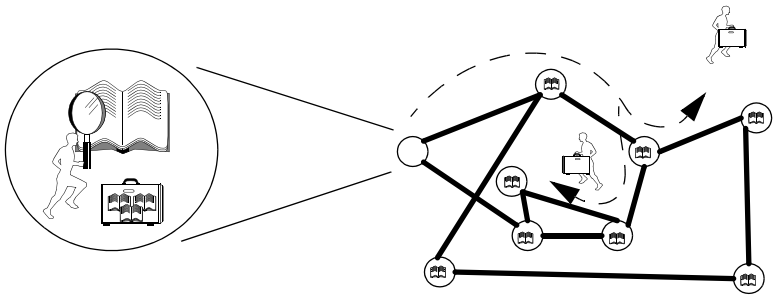

Abb. 3. Agenten durchsuchen das Usenet nach interessanten Artikeln

Natürlich ist eine mobiler Filter-Agent nur dann sinnvoll, wenn die Filterfähigkeiten des mobilen Such-Codes besser sind als die der von der Datenbank angebotenen entfernten Abfrageschnittstelle (sofern es überhaupt eine gibt), und zwar mindestens so viel besser, um den Aufwand der Versendung des Agenten auszugleichen — mit anderen Worten, je raffinierter der Agent ist, desto höher wird der Leistungsgewinn.

Die Anfragesprache, die für die Suchagenten entwickelt wurde, bietet mehr Sprachmittel als Usenet Server und viele andere Datenbestände an ihrer Anfrageschnittstelle anbieten; der volle Vorteil des mobilen Codes kommt allerdings erst zum Tragen, wenn ein Agent beliebige, unvorhersehbare Funktionalität mitbringt — wie etwa ein Suchagent, der eine anwendungsspezifische Analyse der Daten betreibt, z.B. ein Algorithmus zum Textverständnis oder zur Bildanalyse. Als Ansatzpunkt, um diesen charakteristischen Vorteil mobilen Codes ausnutzen zu können, bietet die Suchanwendung die Möglichkeit, eine

benutzerdefinierte Funktion in den mobilen Agenten aufzunehmen; falls vorhanden, wird diese Funktion während der Anfrageverarbeitung aufgerufen, um über die endgültige Einstufung eines Artikels als „interessant“ zu entscheiden. Als einfaches Beispiel für eine solche benutzerdefinierte Filterung wurde eine Funktion zur Verarbeitung von Wortgewichtsvektoren implementiert: Diese Funktion gibt ein positives Resultat zurück, wenn die geeignet gewichtete Summe aller Wörter des Artikels eine numerische „Wichtigkeitsschwelle“ überschreitet. Dieses Beispiel einer benutzerdefinierten Filterfunktion erhöhte die Größe des Agenten um nur 7 KB, was zeigt, dass eine anwendungsspezifische Filterung, die über die Fähigkeiten jeder standardisierten Anfrageschnittstelle hinausgeht, auch unter den engen Beschränkungen der Mobilität möglich und effektiv ist.

Wie in Abb. 4 dargestellt, erreichte die Anwendung bei der Filterung von 6 MB Daten, von denen 20% als interessant herausgefiltert wurden, durch den Einsatz mobiler Agenten Leistungen, die einer äquivalenten stationären Implementierung oft überlegen waren. Der Leistungsvorteil beträgt abhängig vom Durchsatz des Netzwerks etwa 100% bei typischen Schmalband-Geschwindigkeiten von etwa 5 KB/s, und nimmt von dort aus ab bis zu einem Punkt, jenseits dessen die stationäre Variante schneller ist. Dieser Punkt lag dicht am theoretischen Optimum, das durch die lokale Bandbreite des Usenet-Servers (in diesem Fall 90 KB/s) bestimmt wurde. Neben diesen Geschwindigkeitsvorteilen bietet das Suchen mit mobilen Agenten noch den zusätzlichen Vorteil, dass der Client-Rechner abgeschaltet werden kann, solange sich die Agenten noch im Netz bewegen.

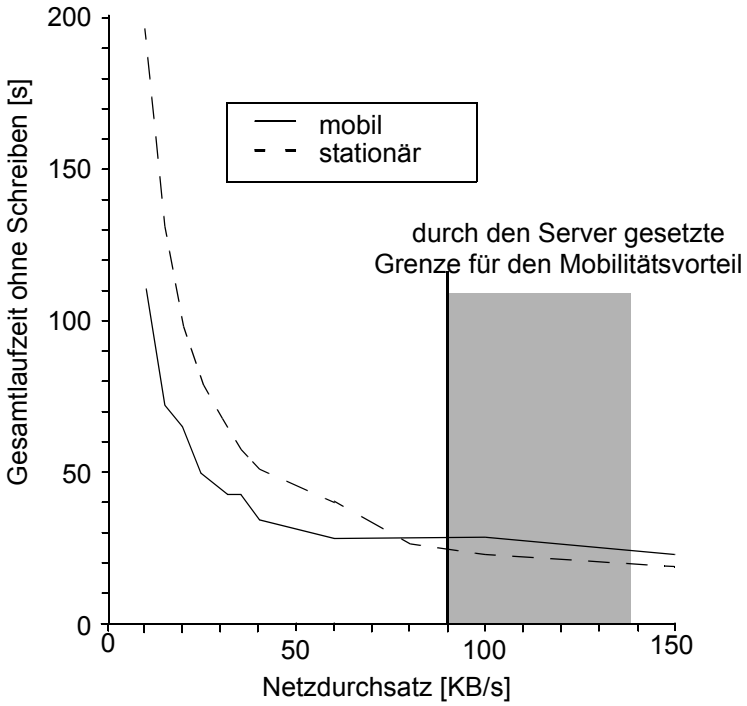

Abb. 4. Mobile und stationäre Laufzeit bei Filterung auf 20% der Originaldaten

Insgesamt hat die Usenet-Suchanwendung gezeigt, dass Geschwindigkeitsvorteile durch mobile Agenten zwar machbar sind, aber im allgemeinen schwer vorhersagbar, da sich herausgestellt hat, dass sie von zahlreichen Parametern abhängen, wie der Leistung der beteiligten Rechner, Netzverbindungen, Mobile-Agenten-Systemen, anderer Software-Komponenten, und natürlich der Anwendung selbst.

5. Zusammenfassung und Bewertung

Mobile Agenten sind ohne Zweifel ein elegantes Konzept der verteilten Programmierung — der ganze Umstand mit der entfernten Kommunikation wird in einem einzigen Migrationsbefehl verborgen. Die Vorstellung eines Agenten, der sich als virtuelles „Alter Ego“ durch das Netz bewegt, ist auch intuitiv sehr ansprechend. Darüberhinaus versprechen mobile Agenten Vorteile hinsichtlich Flexibilität und Geschwindigkeit.

Allerdings bringen mobile Agenten auch ihre ganz eigenen Probleme mit sich, manche davon offensichtlich, wie die Portabilität der Agenten oder die Sicherheit des Wirtssystems, manche weniger offensichtlich, z.B. die Sicherheit des Agenten, der Laufzeitaufwand oder die Kontrolle einer verteilten Agenten-Anwendung. Ara hat gezeigt, dass mobile Agenten unabhängig von der Programmiersprache, mit der komfortablen „starken“ Migration und trotzdem effizient implementiert werden können. *Technisch* gesehen also durchaus erfreuliche Resultate.

Trotzdem fällt die Zukunftsprognose für mobile *Agenten*, im Unterschied zu mobilem *Code*, skeptisch aus. Die Unterschiede zwischen diesen beiden Konzepten wurden in der Vergangenheit oft nicht klar gesehen, und viele Eigenschaften, die schon mobiler Code bietet, wurden oft den mobilen Agenten zugeschrieben, wie die Effizienz der lokalen Interaktion und, langfristig wohl am einflussreichsten, die Flexibilität durch dynamische Rekonfigurierung. Der eigentliche Zusatznutzen von mobilen Agenten gegenüber mobilem Code liegt nicht in der Mobilität der Funktionalität, sondern in der (zusätzlichen) Mobilität der *Kontrolle* über die Funktionsmobilität. Anwendungen, die *diese* zusätzliche Freiheit gewinnbringend nutzen, wurden aber bis heute kaum gefunden.

Aus diesem Grund sieht diese Arbeit eine vielversprechende Zukunft nur für mobilen Code, und zwar vor allem als Baustein in zukünftigen Middleware-Plattformen. Beleg dafür ist die Rolle von mobilem Code in aktueller Middleware wie Enterprise Java Beans, Jini, oder, mehr noch, .NET — Systeme, die die verteilten Anwendungen dieses Jahrzehnts dominieren werden. Auch „bloßer“ mobiler Code benötigt allerdings ein ausgefeiltes Laufzeitsystem für die portable, sichere und effiziente Ausführung, und ein Großteil der Analysen in dieser Arbeit bleibt auch hierfür gültig. Insbesondere Aras sprachunabhängige Architektur passt gut zur prognostizierten Integration von mobilem Code in umfassende Middleware-Plattformen.

Computernetze werden unsere Welt immer weiter und feiner durchdringen, und die Interaktionen im Netz werden immer dynamischer und unerwarteter werden, während Bandbreite auf absehbare Zeit knapp bleibt, wenn man nur an Anwendungen wie mobiles Multimedia und virtuelle oder augmentierte Realität denkt. Die einzigartige Fähigkeit von

mobilem Code, mit unerwarteten Interaktionen und schwachen Netzanbindungen umzugehen, macht dieses Konzept zu einem festen Baustein der Technik, die uns all dies bringen wird.

Literaturverzeichnis

[Pei] Peine, H.: Die Ara-Webseiten. http://wwwagss.informatik.uni-kl.de/Projekte/Ara

[PS97] Peine, H., Stolpmann, T.: The Architecture of the Ara System for Mobile Agents, Proc. of the First International Workshop on Mobile Agents MA'97 (Berlin), 7.-8. April 1997. Lecture Notes in Computer Science Nr. 1219, Springer Verlag, Berlin. ISBN 3540628037.

Nachgedruckt in: Milojicic, D.; Douglis, F.; Wheeler, R. (eds.) : Mobility: Processes, Computers, and Agents, , pp. 474-483, Addison-Wesley and the ACM Press 1999, ISBN 0-201-37928-7.

[Pei97a] Peine, H.: A Plea for Language-Independent Mobile Object Systems, Positionspapier, angenommen für den 3rd ECOOP Workshop on Mobile Object Systems, Jyväskylä, Finnland, 9-10. Juni.

[Pei97b] Peine, H.: Ara - Agents for Remote Action, in W. R. Cockayne and M. Zyda (eds.): Mobile Agents: Explanations and Examples, with CD-ROM, pp. 96 - 164, Manning/Prentice Hall, ISBN 1-884777-36-8 .

[Pei98] Peine, H.: Security Concepts and Implementation for the Ara Mobile Agent System, 7th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 17-19. Juni, Stanford.

[Pei02a] Peine, H.: Application and Programming Experience with the Ara Mobile Agent System. Software -- Practice and Experience (ISSN 0038-0644) 32(6):515-541.

[Pei02b] Peine, H.: Run-Time Support for Mobile Code, Dissertation, Universität Kaiserslautern 2002. ISBN 3925178937. http://wwwagss.informatik.uni-kl.de/Projekte/Ara/Doc/phd-thesis.pdf

[Pei03] Peine, H.: Run-Time Support for Mobile Code and Agents, Künstliche Intelligenz (ISSN 0933-1875), Nr. 2 (April - Juni), S. 56-58.

[Sto95] Stolpmann, T.: MACE - Eine abstrakte Maschine als Basis mobiler Anwendungen, Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern 1995.

Lebenslauf des Autors

Dr. Holger Peine hat das Informatik-Studium an der Univ. Kaiserslautern mit Auszeichnung abgeschlossen und dort als wissenschaftlicher Mitarbeiter über Betriebssysteme, Verteilte Systeme, Rechnernetze und Computersicherheit geforscht. Er hat die Ara-Plattform zur sicheren Ausführung von allgemeinem mobilen Code entwickelt und über dieses Thema bei Prof. Jürgen Nehmer (Vizepräsident der DFG) mit Auszeichnung promoviert und einen Preis für die beste Dissertation der Universität Kaiserslautern gewonnen.

Im Anschluss ist er in die IT-Sicherheitsgruppe am Fraunhofer-Institut Experimentelles Software-Engineering in Kaiserslautern eingetreten, die Werkzeuge für Sicherheitsüberprüfungen von IT-Systemen entwickelt und Sicherheitsevaluationen von Software, Systemen und Prozessen durchführt. Techniken und Werkzeuge für die Entwicklung sicherer Software sind derzeit einer seiner Interessenschwerpunkte.