Erschienen im Tagungsband zum 1.Aachener Arbeitstreffen "Neue Konzepte für die Offene Verteilte Verarbeitung" 5. September 1994, Aachen, ISBN 3-86073-143-2

Trading und Management verteilter Anwendungen: zentrale Aufgaben für zukünftige verteilte Systeme

Ernö Kovacs
Universität Stuttgart, Institut für parallele und verteilte Höchstleistungsrechner (IPVR), Breitwiesenstr. 20-22, 70565 Stuttgart, e-mail: ernoe.kovacs@informatik.uni-stuttgart.de

1.0 Einleitung

Die zukünftige Arbeitswelt wird infolge der Orientierung hin zu flexiblen und anpassungsfähigen Organisationsstrukturen aus kleinen, schnell änder- und anpaßbaren Arbeitseinheiten bestehen. Diese erfordern den verstärkten Einsatz von neuen Informations- und Telekommunikationstechniken um die anstehenden Aufgaben in Büro, Verwaltung, Produktion und Fertigung effizient erledigen zu können. Diese neuen Techniken werden den Nutzern weit über das heutige Maß hinausgehende Kommunikations- und Kollaborationsmöglichkeiten im verteilten System geben. Beispiele hierfür sind multimediale Gruppenkommunikation, synchrone und asynchrone Telekooperationstechniken (z.B. gemeinsam bearbeitbare Objekträume oder eine verteilte Vorgangssteuerung) oder der Einsatz persönlicher Agenten, die im Netz vorhandene Dienste nutzen, um benutzerdefinierte Aufgaben zu erledigen.

Existierende Konzepte für Rechnerkommunikation und verteilte Systeme sind dabei nur sehr bedingt einsetzbar. Einfache Client-Server Systeme unterstützen eine eher statische Verteilungsstruktur, die zwar gewisse Freiheitsgrade bei der Plazierung von Servern (Ortstransparenz) und bei der Zuordnung von Klienten erlaubt, jedoch wenig Möglichkeiten zur dynamischen und flexiblen Anpassung der Systemstrukturen bietet. Es fehlen Möglichkeiten, Systemstrukturen rasch und automatisch an Änderungen anzupassen, sowie auf die Dynamik eines verteilten Systems - beispielsweise auf Änderungen der Last- oder Fehlersituationen - geeignet zu reagieren. Eine Serverauswahl findet meist nur auf Grund eines vordefinierten Kriteriums statt, wobei nur selten eine Optimierung unter Berücksichtigung von statischen und dynamischen Eigenschaften stattfindet. Eine Steuerung der Vielzahl der beteiligten Komponenten und der verteilten Abläufe durch ein Systemmanagement wird nur unzureichend unterstützt.

Um die geforderte Flexibilität und Anpaßbarkeit an sich ändernde Randbedingungen, sowie die Zuverlässigkeit und Kontrollierbarkeit der Systeme zu erlangen, müssen die eingesetzten Entwurfsmethoden und Entwicklungsprinzipien in Hinsicht auf diese Anforderungen weiterentwickelt werden. Weiterhin muß die Überwachung und Steuerung berücksichtigt werden, wobei das Verhalten ganzer Klassen von Objekten des Systems einfach und gezielt angepaßt werden muß. Dies muß jedoch einhergehen mit geeigneten Abstraktionen für die Erstellung solcher Systeme, sowie die Delegation ganzer Aufgabenbereiche an entsprechende Systemfunktionen (beispielsweise Ressourcenmanagement).

Während verteilte objektorientierte Systeme auf der Basis des OMG-Standards ([OMG91]) das Gewicht auf Interoperabilität und Portabilität legt, adressieren die Gebiete Trading ([ANSA91], [ISO92]) und Management verteilter Systeme die Probleme Flexibilität, Anpaßbarkeit und Steuerung des Systems. Mit diesen letztendlich entscheidenden Kriterien für die Akzeptanz eines Systems stellen beide Gebiete zentrale Aufgaben zukünftiger verteilter Systeme dar. Eine nähere Untersuchung zeigt, daß sehr ähnliche Ansätze verfolgt werden. Es

wird jeweils Konfigurationsinformation über das System gesammelt und geeignet eingesetzt. Fehlerinformation wird ausgewertet, korreliert und geeignet berücksichtigt. Der Ablauf eines verteilten Vorgangs wird durch beide Systeme beeinflußt. Es liegt daher nahe, diese Gebiete in einem integrierten Ansatz zu untersuchen.

Der vorliegende Bericht ist wie folgt gegliedert: Im Anschluß an diesen Abschnitt wird auf den Stand der Forschung in beiden Gebieten eingegangen und bestehende Gemeinsamkeiten identifiziert. Dann wird die prinzipielle Vorgehensweise bei der Integration beider Gebiete diskutiert. Im Anschluß daran wird die Architektur des MELODY-Systems vorgestellt und einige Detailaspekte der Integration besprochen. Der Schwerpunkt wird dabei auf die Optimierung bei dem Zugriff auf dynamische Zustandsinformation und der Einsatz dieser Optimierungen im MELODY-Trader gelegt. Der Effekt dieser Optimierungen wird mit Hilfe von Messungen belegt. Am Ende wird eine Zusammenfassung und ein Ausblick gegeben.

2.0 Stand der Forschung

2.1 Trading

Der Gedanke des Tradings entwickelte sich aus der Idee, die Informationen eines Nameservers zu erweitern und damit einen Yellow-Page Dienst zu erbringen. Dieser Auskunftsdienst berücksichtigt vor allen Dingen die Funktionalität und das dadurch geprägte Interface eines Dienstes im verteilten Systems. Er ermöglicht zu einem gegeben Funktionalitäts- oder Interfacespezifikation einen geeigneten Diensterbringer im verteilten System zu finden. Das Trading erlaubt neue Freiheitsgrade bei der Dienstauswahl, indem mit Hilfe einer Subtyp- oder Kompatibilitätsrelation die Auswahlmöglichkeiten auf verschiedene, gleichermaßen verwendbare Diensttypen erweitert wird. Insbesondere in verteilten objektorientierten Systemen können so auch Objekte einer abgeleiteten Klasse gefunden werden. Diese Möglichkeit erlaubt die gezielte Weiterentwicklung von Diensten unter Beibehaltung einer Aufwärtskompatibilität ([ANSA91]).

Eigenschaften eines Dienstes werden mit Hilfe von beschreibenden Attributen ausgedrückt. Diese können bei dem Trading-Vorgang berücksichtigt werden und im Auswahl- und Optimierungsprozeß eingesetzt werden. Bei Attributen wird zwischen statischen und dynamischen Attributen unterschieden. Dynamische Attribute ändern ihren Wert im Lauf der Zeit und erfordern daher geeignete Maßnahmen auf die neue Information zuzugreifen oder die Änderungen zu propagieren. Die Leistung eines Tradingsystems kann durch eine effiziente Gestaltung dieser Zugriffe sehr gesteigert werden.

Zur Strukturierung des Dienstangebots nach beliebigen organisatorischen, administrativen oder individuellen Kriterien, werden Dienstangebote in Trading-Directories - dem Kontextbaum (vergleichbar einem File-Directory) - eingetragen. Die Hierarchie dieser Directories kann beliebig gestaltet werden. Beispielsweise kann eine geographische Lokalisation von Diensten in der Hierarchie wiedergegeben werden. Die Directory-Hierarchie wird genutzt, um den Suchbereich für Dienste einzuschränken.

Bei der Gestaltung des Tradingraums wird von einem einheitlichen, hierarchischen Dienstraum (wie er z.B. in einem X.500 System besteht) abgesehen. Es werden getrennte Tradingdomänen vorgesehen, innerhalb derer ein einheitliches Tradingsystem etabliert ist. Tradingdomänen können auf unterschiedliche Arten zusammenarbeiten. Bei föderativen Tradern ([BeRa91]) wird ein Trading-Kontrakt etabliert, der definiert, welche Diensttypen aus einer Tradingdomäne in einer anderen sichtbar sind, welche Abbildungen von Diensttypen dabei vorgenommen werden müssen und welche Teile des jeweiligen Dienstraumes exportiert werden.

2.2 Management verteilter Systeme

Das Management verteilter Systeme beschäftigt sich mit der Beobachtung und Steuerung verteilter Systeme. Das verteilte System wird dabei als eine Sammlung von *Managed Objects* (MOs) aufgefaßt, die jeweils eine reale Ressource repräsentiert- beispielsweise ein Gerät der Netzinfrastruktur, ein Systemobjekt oder eine Anwendungskomponente. Eine Manipulation des MOs wirkt auf die reale Ressource ein. Diese Manipulationen werden durch geeignete Managementfunktionen überwacht und gesteuert. Managementfunktionen werden durch *Managing Objects (MngOs)* erbracht.

Zur Reduzierung der mit der Vielzahl von anfallenden Managementaufgaben verbundenen Komplexität, werden Managementaufgaben häufig nach *Funktion*, *Lebensbereich* und *Zielobjekt* kategorisiert. Die Funktion kann weiter in die fünf Bereiche *Konfiguration*, *Fehlerbehandlung*, *Leistungsmanagement*, *Abrechnung* und *Sicherheit* unterteilt werden. Innerhalb der Kategorie Lebensbereich werden die Phasen *Planung*, *Entwurf*, *Installation*, *Betrieb* und *Migration* unterschieden, bei den Zielobjekten hingegend die Bereiche *Netzwerk*, *(End-) System* und *Anwendungen*.

Entwicklungen im Bereich Netzwerk- und Systemmanagement haben bereits Produktstatus erlangt. Hier gibt es mit SNMP ([Rose91]) und den OSI-Managementprotokollen auch eine standardisierte Basis. Für den Bereich Management einer verteilten Anwendung existieren jedoch nur wenig Erfahrungen. SNMP dürfte sich hier als zu starr und unflexibel erweisen. Die objektorientierten Ansätze des OSI Management sind eine ausreichende Basis, jedoch fehlt es noch an konkreten Erprobungen und geeigneten Entwicklungsunterstützung. Ansätze wie das Object Management Framework des DME ([OSF91]) beruhen auf dem CORBA-Standard, haben jedoch noch keine konkreten Ergebnisse erzielen können. Allgemein wird SNMP als defacto Standard akzeptiert, Lösungen auf der Basis des OSI- oder des DME-Framework als die Zukunft angesehen. Proprietäre Lösungen werden noch auf lange Jahre gute Chance am Markt haben.

Das Management großer verteilter System muß ebenfalls verteilt sein, um mit der Größe des Systems umgehen zu können, sowie um eine Überlastung der Managementstation und der Netzwerke zu vermeiden. Hier bietet sich eine hierarchische Dekomposition der Managementfunktionen an. Einzelne Funktionen dieser Dekomposition können dann auf verschiedenen Knoten des Netzes ausgeführt werden. Übergeordnete Managementfunktionen führen die Ergebnisse der untergeordneten Funktionen zusammen. Interessanterweise kann ein Trader bei der Auswahl eines geeigneten Ausführungsorts für eine Managementfunktion eingesetzt werden. Hilfestellung kann dabei die Strukturierung von MOs in (hierarchisch strukturierte) *Managementdomänen* ([SMTK93]) sein. Mit Hilfe dieser Strukturierung können Gruppen von MOs gebildet werden. Eine *Managementpolitik* ([Mars93]) definiert für eine Gruppe von MOs eine gemeinsames Verhaltensmuster. Mit Hilfe von Politiken können große verteilte Systeme einheitlich verwaltet werden. Eine Änderung einer Politik muß dann eine Verhaltensänderung bei allen betroffenen Objekten nach sich ziehen.

Die Forschung auf den Gebieten Management verteilter Anwendungen, verteiltes Management, Managementdomänen und Managementpolitiken ist jedoch noch nicht sehr weit fortgeschritten, so daß akzeptierte und vereinigende Ansätze fast vollständig fehlen.

2.3 Gemeinsamkeiten

Beide Gebiete haben eine Reihe von Gemeinsamkeiten, die eine integrierte Untersuchung erforderlich machen. Eine Reihe Gemeinsamkeiten wird im folgenden aufgezählt:

Konfigurationsinformation
Auf der Basis der exportierten Konfigurationsinformation selektiert der Trader geeignete Diensterbringer. Beim Management verteilter Systeme wird Konfigurationsinformation von Managementfunktionen gesammelt und geeignet gespeichert. Sie bildet dann die Grundlage für

weitere Managementfunktionen, beispielsweise der Fehlersuche oder der Präsentation für einen Benutzer.

Fehlerinformation
Der von einem Trader vermittelte Dienst wird im Regelfall im Anschluß daran von einem Klienten angefordert. Daher sollte aus Effizienzgründen der Trader den Zustand eines Servers überprüfen können und nicht funktionierende Server bei der Vermittlung übergehen. Eine der Hauptaufgaben des Fehlermanagement ist die Identifikation von aufgetretenen Fehlern, indem Ereignisse aus dem System gesammelt, gefiltert und korreliert werden. Aufgetretenen Fehlern kann dann durch geeignete Maßnahmen entgegengewirkt werden.

Leistungsdaten
Ein Trader verwendet bei der Auswahl eines optimalen Dienstes häufig Leistungsdaten über einen Server, die sich entweder statisch aus prinzipiellen Möglichkeiten eines Servers oder dynamisch aus sich ändernden Leistungsfaktoren ergeben. Das Leistungsmanagement beschäftigt sich mit der Ermittlung von Leistungszahlen über ein Objekt des verteilten Systems. Auch hier können statische Leistungszahlen (beispielsweise die MIPS-Zahl eines Rechenknotens) oder dynamische Leistungsfaktoren ermittelt werden. Häufig werden Leistungsdaten auch durch langfristige Beobachtungen und Trendanalysen gewonnen (Beispiel: durchschnittliche Auslastung).

Kosten
Die Kosten einer angeforderten Dienstleistung können ein wesentlicher Faktor bei der Auswahl eines Dienstes sein. Ein Trader muß daher über die Tarife eines Dienstes und einzelner Operationen informiert sein. Das Abrechnungsmanagement beschäftigt sich mit genau dieser Gestaltung von Tarifen und mit der Sammlung von Abrechnungsinformation.

Sicherheit
Ein Trader sollte nur Dienste vermitteln, die für einen Klienten auch verfügbar sind. Es ist daher wichtig, daß bei der Dienstauswahl auch die Überprüfung der Autorisierungsrechte erfolgt. Dies muß natürlich eng mit dem Sicherheitsmanagement zusammenarbeiten, welches die Autorisierungsrechte verwaltet.

Aus den aufgezählten Beispielen läßt sich unschwer erkennen, daß ein enger Zusammenhang zwischen einem Tradingsystem und dem Managementsystem für die verteilten Dienste besteht. Ein integrierter Ansatz bei der Behandlung beider Systeme ist daher erforderlich.

3.0 Integrationskonzepte

Es gibt eine Reihe von unterschiedlichen Konzepten, wie Trading und Management integriert werden können:

Vollständige Integration
Bei der vollständigen Integration wird das Trading als eine spezielle Managementfunktion aufgefaßt, woraufhin Interaktionen mit dem Trader dann mit Hilfe von Managementprotokollen erfolgen. Dieser Ansatz hat den Vorteil, daß sowohl im Trader als auch in dem Managementsystem ein einheitliches Informationsmodell verwendet wird. Es sind daher keine aufwendigen Abbildungsschritte notwendig. Managementframeworks nach dem OSI- oder CORBA-Standard erlauben eine vollständige Abbildung der Trader-Schnittstelle auf das Interface eines Trader-MOs. Es muß sich jedoch zeigen, ob eine solche Lösung dem Traderproblem angemessen ist. Weiterhin erzwingt diese Lösung die vollständige Ablösung bestehender Trading-Systeme und den Ersatz durch ein entsprechendes Trader-MO.

Hierarchische Integration
Der Trader bildet eine Anwendung, die auf Managementinformation zugreift. Diese Lösung ermöglicht es, das Managementsystem unabhängig von dem Trader zu halten. Es besitzt eine

einfache und klare Struktur. Der Trader muß das Informationsmodell des Managementsystems auf das eigene Informationsmodell abbilden. Die Schnittstelle des Managementsystems muß derart gestaltet werden, daß es die benötigte Managementinformation effizient dem Trader zur Verfügung stellen kann. Aus der Sicht des Traders ist dies eine ausreichende Lösung. Jedoch werden hierbei die Möglichkeiten des Traders zur Steuerung des Systems nicht ausgenutzt. Beispielsweise kann die Auswahl eines Dienstes durch das Systemmanagement beeinflußt werden. Dazu müßte das Management auf den Trader zugreifen können, was bei eines strikt hierarchischen Integration nicht möglich ist.

Gleichberechtigte Integration
Bei der gleichberechtigten Integration stellen beide Systeme eine Schnittstelle zur Verfügung, über die das andere zugreifen kann. So kann der Trader sich über die Fehlerinformation des Managementsystems informieren, während das Managementsystem den Auswahlprozeß durch die Definition geeigneter Regeln (Politiken) beeinflussen kann. Während der Instanziierung einer Managementfunktion kann durch den Trader eine geeignete Ausführungsumgebung ermittelt werden, u.v.m. Es muß jedoch sehr stark darauf geachtet werden, daß die wechselseitigen Nutzungen nicht zu Deadlock-Situationen führen.

Die vollständige Integration vereinigt beide System auf der Basis gemeinsamer Kommunikationsprotokolle und einem einheitlichen Informationsmodell. Auf Grund der bereits bestehenden getrennten Entwicklungen wird dies in der Praxis selten der Fall sein. Die hierarchische Integration ist ein einfaches Modell, welches für den Trader durchaus ausreichend sein mag. Es berücksichtigt jedoch die Anforderungen des Management nur unzureichend. Eine gleichberechtigte Integration birgt die Problematik, daß die Interaktionen zwischen den System sehr sorgfältig entworfen werden müssen, um Verklemmungen und gegenseitige Abhängigkeiten zu vermeiden. Es ergeben sich aber viele Möglichkeiten, die Ansätze beider Gebiete gemeinsam zu nutzen.

4.0 Die MELODY-Architektur

Das MELODY-System (Management Environment for Large Open Distributed sYstems) beinhaltet eine Trader-Komponente und ein verteiltes Managementsystem für verteilte Anwendungen. Trader und Managementsystem sind in erster Linie hierarchisch integriert, d.h. der Trader nutzt die Möglichkeiten des Managementsystems, um

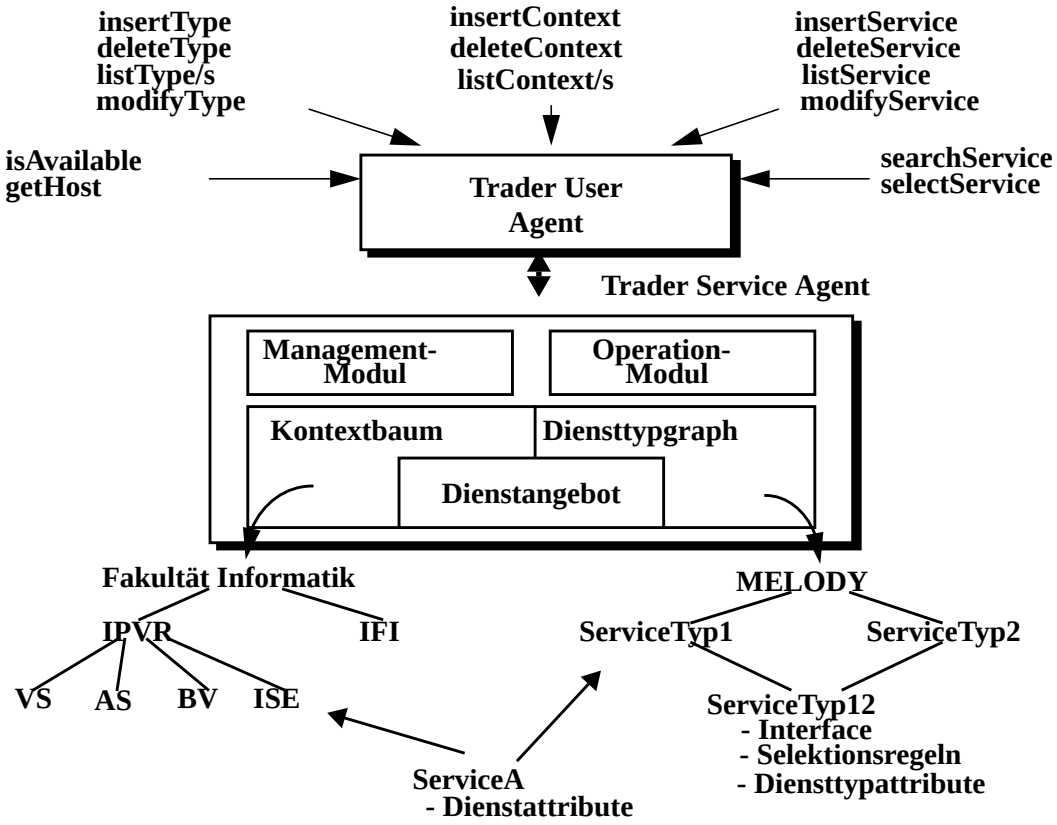
- auf dynamische Zustandsinformation von Anwendungskomponenten zuzugreifen,
- die Verfügbarkeit eines Dienstes zu überprüfen,
- über aufgetretene Fehler informiert zu werden,
- über kontinuierliche Eigenschaften eines Dienstes (beispielsweise prozentuale Verfügbarkeit) informiert zu sein
- sich über Umgebungs-Eigenschaften eines Dienstes (z.B. Systemlast, Netz-Roundtripzeiten zwischen Klient und Server, u.v.m.) zu informieren.

Im Trader sind jedoch auch Vorkehrungen getroffen, um Managementpolitiken durchzusetzen. So können im Trader Regeln gespeichert werden, die bei einer Dienstauswahl durch den Trader direkt angegeben, indirekt durch den Trader durchgesetzt werden oder als DEFAULT-Auswahlkriterium gelten. Hier werden erste Ansätze für eine gleichberechtigte Integration sichtbar. Im folgenden wird die Trader-Architektur vorgestellt, bevor auf die Eigenschaften des Managementsystems eingegangen werden.

4.1 Die Architektur und das Informationsmodell des Traders

Der MELODY-Trader (Wira94) besteht aus einem Trader User Agent (TUA) und einem Trader Service Agent.(TSA). Der TUA kommuniziert mit dem TSA. Der TSA führt die eigentli-

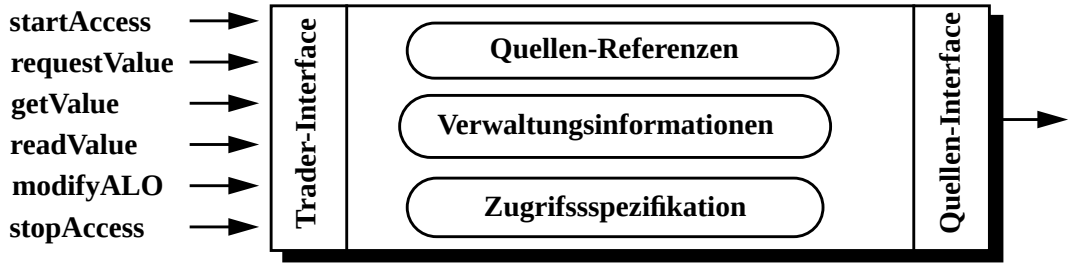
chen Trader-Operationen aus. Er speichert den Kontextbaum, den Diensttyp-Graphen und die Dienstangebote. Ein Operationsmodul führt auf den Trader-Datenstrukturen die Traderoperationen aus. Das Managementmodul stellt Managementinformation über den Trader bereit und kommuniziert mit dem Managementsystem.



Der TUA bietet die Operationen des Traders einem Tradernutzer (Klient, Server oder Systemmanager) an und wird zu einem entsprechenden Programm hinzugelinkt. Es existieren ein graphisches und ein textbasiertes Interface, mit dem die entsprechenden Traderoperationen direkt aufgerufen werden können.

Dienstangebot und Dienstattribute

Ein Dienstangebot besteht aus dem Namen des Dienstes, dem Diensttyp und den Dienstattributen. Dienstattribute können aus verschiedenen Quellen stammen. Beispielsweise sind statische Attribute im Trader gespeichert, während auf dynamische Attribute mit Hilfe des MELODY-Managementsystems zugegriffen werden muß. Andere Zugriffsarten, beispielsweise auf persistente Information in einer Datenbank, sind denkbar. Um diese Zugriffsheterogenität zu verbergen, wird das *Access Location Object (ALO)* eingeführt, das eine abstrakte Zugriffsmöglichkeit auf den Wert eines Attributs bietet. In Abhängigkeit von der Art des Attributs wird ein von einem ALO abgeleitetes Objekt instanziiert, welches die geforderte Zugriffsart verwendet.



Zugriffsschemata

Bei einigen Attributen hängt der Zugriff noch von dem Wert anderer Variablen ab, beispiels-

weise dem Ort des Diensterbringers. Hierzu können durch ein *Access Location Template (ALT)* Zugriffsschemata definiert werden. Beispielsweise ist die Systemlast ein Dienstattribut, das durch das ALT definiert wird. Diese ALT enthält eine Variable für den Rechner, auf dem der Dienst läuft. Beim Zugriff auf ein Dienstattribut werden die ALT-Variablen im Kontext des konkreten Dienstes evaluiert. Beim Beispiel der Systemlast wird bei der Verwendung des Attributs dynamisch der Rechner bestimmt, auf dem der konkrete Dienst läuft und dann auf dessen Systemlast zugegriffen.

Selektions- und Optimierungskriterien
Bei der Dienstauswahl kann ein Auswahl- und ein Optimierungskriterium angegeben werden. Das Auswahlkriterium ist ein boolscher Ausdruck, wogegen das Optimierungskriterium eine Vergleichszahl zur Ordnung der Dienste liefert. Beide Ausdrücke beinhalten Dienstattribute oder Zugriffsschemata. Selektions- und Optimierungskriterien können im Trader gespeichert werden und mit Hilfe eines Namens referenziert werden. Dies vereinfacht z.B. dem Klienten die Angabe von Auswahlkriterien, da er nur einen definierten Namen und keinen Ausdruck angeben muß. Durch die Indirektion kann auch das Systemmanagement unabhängig von den Klienten die Auswahlkriterien beeinflussen. Neben diesen expliziten Auswahlregel, gibt es noch DEFAULT-Regeln, die beim Diensttyp angegeben werden können. Diese werden ausgewählt, wenn der Klient keines explizites Auswahlkriterium angibt. Weiterhin gibt es noch implizite Auswahlregel, die im Trader gespeichert sind. Diese werden implizit zu einem vorgegebenen Kriterium hinzugefügt.

4.2 Das MELODY-Managementsystem

Das MELODY-Managementsystem besteht aus einem Management-Agenten (MMA), der auf jedem Systemknoten läuft. Komponenten einer Anwendung (oder eines Dienstes) melden sich beim MMA an und stellen ihre Managementinformation zur Verfügung. Eine Managementanwendung greift mit Hilfe ihres lokalen MMA auf Managementinformation zu. Hierzu kommunizieren die MMAs verschiedener Systeme.

Managed Objects (MOs)
MOs repräsentieren die Managementinformation einer Anwendung. Sie besitzen einen globalen Namen, beschreibende Attribute, eine Reihe von möglichen Events und Aktionen. Der globale Name identifiziert das MO eindeutig und läßt sich zu der Knotenadresse des MOs auflösen. Attribute können synchron oder asynchron gelesen und geschrieben werden. Genauso kann die Strukturinformation eines MOs (Anzahl und Typ der Attribute, Anzahl, Name und Parametertyp der Events, Anzahl, Name und Parametertyp der Aktionen) dynamisch gelesen werden.

Das MELODY-Managementsystem unterstützt das Anlegen von Kopien von entfernten MOs auf dem eigenen Knoten (Schattenobjekte). Diese Schattenobjekte sind eine inkonsistente Kopie des Originalobjekts. Das Managementsystem sorgt dafür, daß vorgegebene Aktualitätsanforderungen beim Zugriff auf das Schattenobjekt eingehalten werden ([Kova93], [Helb92]). Dabei wird dynamisch eine optimale Zugriffsstrategie ausgewählt (Direct Access oder Reporting).

Managing Objects (MngO)
MngO sind Objekte, die eine Managementfunktion ausführen. Beispielsweise kann eine MngO die Verfügbarkeit einer laufenden Anwendungskomponente überwachen. MngO können dynamisch auf sogenannten Object Servern instanziiert werden und berechnen abgeleitete (indirekte) Information über eine Anwendungskomponente. MngO können auch Informationen über das jeweilige System, über die Verzögerung zwischen zwei Knoten, u.v.m. liefern.

4.3 Zusammenarbeit zwischen Trader und Managementsystem

Dynamische und indirekte Attribute
Das Managementsystem stellt die dynamischen und die indirekten Attribute eines Dienstes zur

Verfügung, indem mit dem Dienstangebot die entsprechenden Angaben über die Beziehung zwischen einem Dienstattribut und einem Managementattribut exportiert werden. Weiterhin kann der Trader MngO instanziieren, um eine Anwendungskomponente zu überwachen und um beobachtete (indirekte) Information über die Komponente zu erhalten. Die Vorgehensweise hierbei folgt strikt der hierarchischen Integration, da das Tradingsystem das Managementsystem verwendet.

Die Auswertung eines Auswahlkriteriums erfolgt in zwei Stufen. In der ersten Stufe werden die Teilausdrücke bewertet, die nur statische Attribute besitzen. Hat sich dann kein Ergebnis ergeben, so erfolgt der Zugriff auf dynamische Attribute. Der Zugriff kann in unterschiedlicher Weise erfolgen:

- Synchron - Jeder Attributzugriff wird synchron durchgeführt
- Asynchron - Alle Attributzugriffe eines Dienstes werden angestoßen, bevor auf die Ergebnisse gewartet wird.
- Parallel - es stehen eine Anzahl Threads bereit, die jeweils ein Dienstangebot auswerten.

Diese Ansätze reduzieren die Antwortzeiten von Anfragen mit dynamischen Parametern erheblich. Jedoch besteht weiterhin das Problem, daß bei einer großen Menge an zutreffenden Diensten viele Netzzugriffe und somit ein hohe Kommunikationsaufwand entsteht. Eine weitere Reduzierung kann durch ein lokales Cachen der Information erreicht werden. Dabei müssen jedoch unterschiedliche Eigenschaften der Attribute (z.B. Änderungsfrequenz, die Wichtigkeit akkurater Information) berücksichtigt werden.

Fehlermeldungen
Das Managementsystem kann Ereignisnachrichten direkt an den Trader weitergeben. Diese informieren ihn z.B. über den Absturz eines Servers. Dieser kann dann als nicht verfügbar markiert werden und bei späteren Vermittlungen nicht berücksichtigt werden.

Auswahlregel
Im Trader können durch das Systemmanagement Auswahlregeln gespeichert werden, die bei der Dienstauswahl referenziert werden. Durch diese Indirektion kann das Systemmanagement Einfluß auf die gewählte Auswahlstrategie nehmen. Mit Hilfe von impliziten und DEFAULT-Regel können weitere Verhaltensmuster für die Dienstnutzung erzwungen werden.

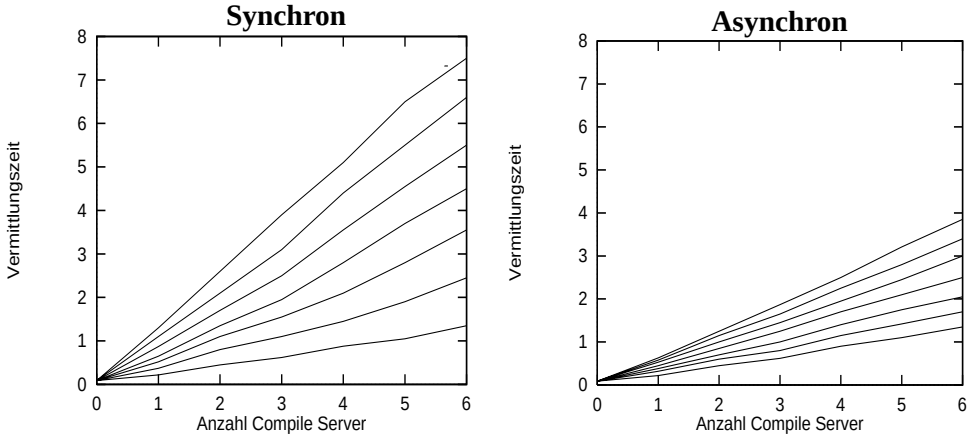
Ressourcenmanagement
Der MELODY-Trader kann die Möglichkeiten des Managementsystems nutzen, um in einem Server Reservierungen für den Klienten vorzunehmen. Implementiert wurde eine einfache, auf einem Timeout-basierende Reservierungsstrategie.

5.0 Leistungsmessungen

Die folgenden Leistungsmessungen wurden anhand eines Beispielszenarios durchgeführt bei dem ein Compiledienst durch den Trader vermittelt wurde. Als dynamische Attribute dienten die Anzahl der Compile-Jobs in der Warteschlange, die bisherige mittlere Bearbeitungszeit, die Systemlast des zugehörigen Knotens (ein indirektes Attribut), u.a. Messungen wurden mehrfach durchgeführt und Extremwerte eliminiert. Die Last auf den einzelnen Rechnern, sowie auf dem Netz wurde minimiert. Gemessen wurde auf IBM RS/6000-Workstations unter AIX, die durch ein Ethernet verbunden sind.

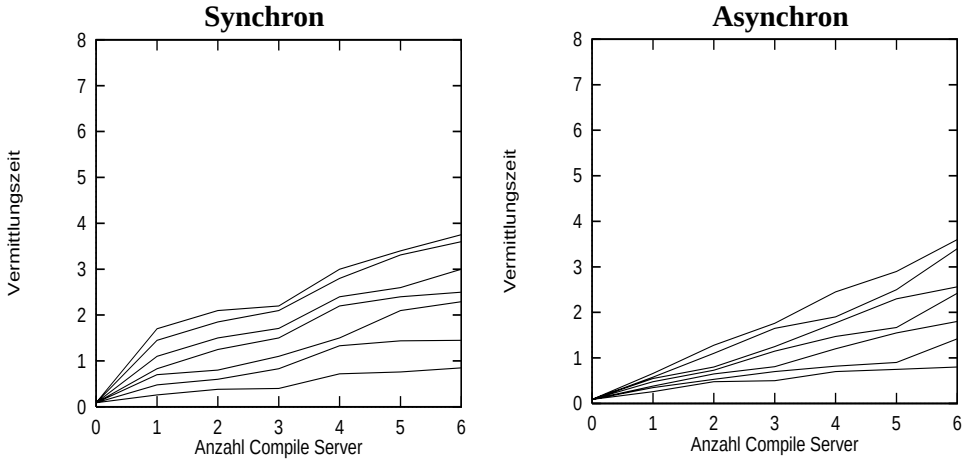
Die Ergebnisse demonstrieren das Systemverhalten für die folgenden Einstellungen:

- Keine Aktualitätsprädikate / keine Parallelität:



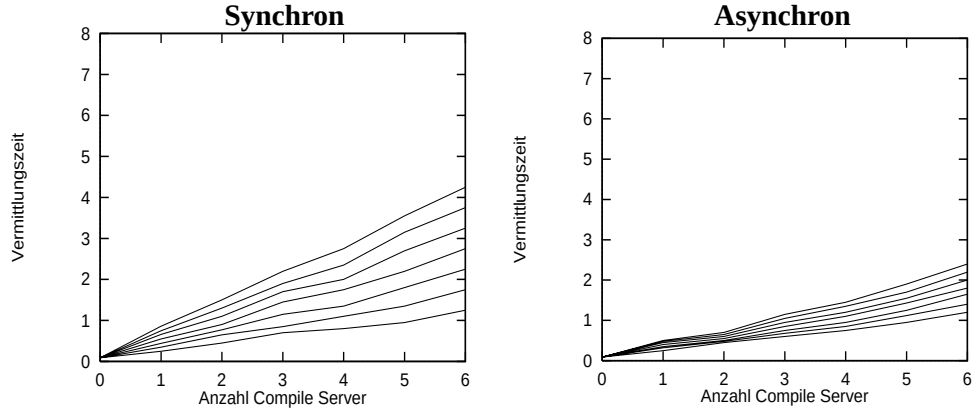
Unterschiedliche Linien in dem Diagram zeigen eine steigende Anzahl von dynamischen Parametern. Der asynchrone Zugriff führt zu einer deutlichen Verbesserung der Vermittlungszeiten. Wie zu erwarten bleibt der Aufwand bei nur einem dynamischen Attribut gleich.

- Keine Aktualitätsprädikate / Parallelität:



Innerhalb des Traders existiert eine vordefinierte Anzahl von Threads. Die Berechnung der Auswahlkriterien für einzelne Dienstangebote wird jeweils einem Thread übergeben. Durch die parallele Auswertung ergeben sich dann kaum noch Unterschiede zwischen der synchronen und der asynchronen Auswertung, da die Wartezeiten auf Rückantworten durch parallel arbeitende Threads genutzt wird. Die Knicke in den Kurven lassen sich durch die Anzahl der eingesetzten Threads (in der Meßreihe drei) begründen..

- Aktualitätsprädikate / keine Parallelität



In diesem Beispiel wurden für die eingesetzten dynamische Attribute sinnvolle, aber unterschiedliche Aktualitätsprädikate gesetzt. Da es bei Aktualitätsprädikaten auch auf die Frequenz der Zugriffe ankommt, damit der Cache gefüllt ist, wurde vor den Messungen erst eine Einschwingphase vorneangestellt, die den Cache gut füllt.

6.0 Zusammenfassung und Ausblick

Das vorliegende Paper motiviert die Bedeutung von Flexibilität und Systemkontrolle für zukünftige verteilte Systeme. Es zeigt auf, daß die beiden Forschungsgebiete Trading und Management verteilter Systeme zentrale Themen für diese Aufgabenstellung sind. Es führt in die aktuelle Forschung ein und zeigt die engen Querbezüge auf. Es werden Ansätze zur Integration besprochen und eine Realisierung anhand des MELODY-Systems gezeigt. Dabei werden einige Detailprobleme herausgegriffen, genauer erläutert und die gefundenen Lösungen durch Messungen bestätigt.

Bei den Messungen muß die Skalierbarkeit der Lösungen noch nachgewiesen werden. Weiterhin läßt sich erwarten, daß sich die Zusammenarbeit zwischen Managementsystem und Trader noch weiter verbessern läßt - z.B. durch eine bessere Zusammenarbeit beim Starten von Diensten, durch Verbesserungen bei der Fehlertoleranz oder durch bessere Ressourcenmanagementstrategien. Hier ergeben sich noch eine Reihe von offenen und interessanten Fragestellungen.

Literatur:

[ANSA91] ANSA. ANSAware 3.0 Implementation Manual. Manual RM.097.01, Architecture Projects Management Limited, February 1991.

[BeRa91] Mirion Bearman, Kerry Raymond. Federating Traders: an ODP Adventure. In Jan de Meer, Volker Heymer (Hrsg.), *Proceedings of the International IFIP Workshop on Open Distributed Processing*, 1991.

[Helb92] Tobias K. Helbig. Strategien zur Einhaltung von Aktualitätsanforderungen. Diplomarbeit Nr. 944, Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner, Stuttgart, Dezember 1992.

[ISO92] ISO. Working Document on Topic 9.1 - ODP Trader. *Working paper of the ISO/IEC JTC1/SC21/WG7: N7047*, May 1992.

[Kova93] Ernö Kovacs. Automatic Selection of an Update Strategy for Management Data. In *Proceedings of the IEEE First International Workshop on Systems Management (IWSM'93)*, Los Angeles, April 1993.

[Mars93] Lindsay F. Marshall. Representing Management Policy Using Contract Objects. In *IEEE First International Workshop On Systems Management*, Los Angeles, April 1993. IEEE.

[OMG91] OMG. The Common Object Request Broker: Architecture And Specification. Technischer Bericht 91.12.1, Object Management Group, December 1991.

[OSF91] OSF. The OSF Distributed Management Environment - A White Paper. Technischer Bericht, OSF, January 1991.

[Rose91] Marshall T. Rose. *The simple book : an introduction to management of TCP-IP -based internets*. Prentice-Hall series in innovative technology. Prentice Hall, Englewood Cliffs, NJ, 1991. XXIX, 347 S.

[SMTK93] M. Sloman, J. Magee, K. Twidle, J. Kramer. An Architecture For Managing Distributed Systems. In *Proceedings of the 4th Workshop on Future Trends in Distributed Systems*, S. 40–46. IEEE, 1993.