in: Proc. 6. GI/ITG-Fachtagung "Messung. Modellierung und Bewertung von Rechensystemen", München, Springer-Verlag, Informatik-Fachberichte 286, Sept. 1991, S. 61-75.

Messen und Bewerten paralleler Client/Server-Architekturen - am Beispiel des kooperierenden Non-Standard-Datenbanksystems PRIMA

Michael Gesmann, Christoph Hübel, Wolfgang Käfer, Harald Schöning, Bernd Sutter
Universität Kaiserslautern, Fachbereich Informatik
Postfach 3049, 6750 Kaiserslautern

Zusammenfassung

Mit der stetigen Verbreitung und Vernetzung von Arbeitsplatzrechnern sowie dem wachsenden Einsatz von Mehrrechnersystemen gewinnen auftragsorientierte Systemmodelle, die eine Zerlegung in auftraggebende und auftragnehmende Systemkomponenten vorsehen, zunehmend an Bedeutung. Die effektive Nutzung der verteilten Rechnerleistung erfordert geeignete Maßnahmen zur Unterstützung der parallelen Auftragsabwicklung. Neben Konzepten zur Betriebssystemeinbettung adäquater Ablaufumgebungen existiert ein Bedarf an Meßwerkzeugen und Analysemethoden, die eine quantifizierbare Bewertung paralleler, auftragsorientierter Anwendungssysteme erlauben.

Am Beispiel des PRIMA-Systems, einer Prototypentwicklung eines kooperierenden Nicht-Standard-Datenbanksystems, wird ein konkretes auftragsorientiertes Systemmodell vorgestellt und die Meß- und Analyseproblematik verdeutlicht. Dabei werden eine Reihe speziell entwickelter Meß- und Analysewerkzeuge erläutert und ein Bewertungsvorgehen diskutiert. In einer Fallstudie werden die Auswirkungen einer praktisch durchgeführten Systembewertung auf die Gestaltung einer einzelnen PRIMA-Komponente demonstriert.

Schlüsselwörter

Client/Server-Architektur, Konfiguration, Meßmethoden, Leistungsbewertung, ereignisgesteuerte Software-Messung

1. Einleitung und Problembeschreibung

Zerlegung und Verteilung stellen Grundprinzipien bei der Lösung und Beherrschung algorithmischer Problemstellungen sowie bei der Erstellung komplexer Anwendungssysteme dar. Bei der Systemmodellierung gewinnen funktions- oder auftragsorientierte Systemmodelle zunehmend an Bedeutung. Diese beruhen auf der Zerlegung eines Anwendungssystems in Komponenten, die durch gegenseitige Erteilung von Aufträgen die Funktionalität des Gesamtsystems realisieren. Bei der Systemgestaltung kann daher oftmals sehr natürlich die operationale Zerlegung, also das Unterteilen einer komplexen Operation in eine Folge einfacherer Teiloperationen, berücksichtigt werden. Ist die Abarbeitung der Teiloperationen voneinander unabhängig, kann durch verschiedene Systemkomponenten eine verteilte Operationsabwicklung erfolgen. Stehen hierfür mehrere physische Ausführungseinheiten (Prozessoren) zur Verfügung, so ist eine parallele Abarbeitung möglich. Die Wahl einer geeigneten Systemkonfiguration bildet damit ein wichtiges Optimierungspotential [Zi91].

Neben der Schaffung geeigneter Ablaufumgebungen, die eine Einbettung solcher auftragsorientierter Systemmodelle in einer verteilten und ggf. heterogenen Betriebssystem- und Hardware-Umgebung erlauben, müssen die Durchführung von Messungen sowie die Analyse und die Bewertung konkreter Anwendungssysteme durch entsprechende Werkzeuge unterstützt werden [Br88, Mi90]. Dies bereitet u.a. aufgrund der verteilten Systemumgebung erhebliche Probleme. So werden Meßwerte zwangsweise dezentral erfaßt, wodurch die Zusammenfassung und Korrelation, die für eine komponentenübergreifende Interpretation erforderlich ist, ein eigenständiges Problem darstellt. Weiterhin existieren eine Vielzahl von äußerst heterogenen Parametern, die die Systemeinbettung und die Systemkonfiguration bestimmen und somit einen entscheidenden Einfluß auf die Systemdynamik und das gesamte Leistungsverhalten besitzen. Darüber hinaus ist die Frage nach den insgesamt relevanten Leistungsmerkmalen und nach den geeigneten Bewertungsmaßen und -modellen zu klären.

In dem hier vorliegenden Beitrag werden Bewertungskonzepte und -techniken für auftragsorientierte, dezentral und parallel ablaufende Anwendungssysteme vorgestellt und an einem durchgängigen Beispiel aus dem Bereich der Datenbankentwicklung motiviert und verdeutlicht. Die erläuterten Konzepte bieten gerade für den Entwicklungsbereich ein wichtiges Hilfsmittel bei der Optimierung und Anpassung von Systemkonfigurationen, bei der Überprüfung von Entwurfsentscheidungen sowie bei der Bewertung und Validierung von Implementierungskonzepten.

Im folgenden wird am Beispiel des PRIMA-Systems [HMMS87], eines Nicht-Standard-Datenbank-Kernsystems (NDBS) für technische Anwendungen, zunächst ein konkretes auftragsorientiertes Systemmodell sowie eine allgemeine Ablaufumgebung für verteilte, parallele Client/Server-Systeme vorgestellt. Kapitel 3 befaßt sich im Anschluß daran mit Fragen nach den zu variierenden Systemparametern sowie nach den zu erfassenden, relevanten Meßgrößen. In Kapitel 4 werden dann eine Reihe konkreter Möglichkeiten für die Gewinnung von Meßdaten sowie mögliche Vorgehensweisen bei der Parametervariation und der Systembewertung diskutiert und entsprechende Meß- und Analysewerkzeuge beschrieben. Abschließend werden erste praktische Erfahrungen mit der Meßmethodik anhand des PRIMA-Systems skizziert und die Auswirkungen der durchgeführten Systembewertung auf eine verbesserte Systemgestaltung aufgezeigt.

2. Realisierung einer parallelen Client/Server-Architektur für das NDBS PRIMA

2.1 Das Client/Server-Modell

Beim Client/Server-Ansatz wird ein komplexes Anwendungssystem in eine Reihe von ***Anwendungskomponenten*** zerlegt, deren Zusammenspiel nach dem *Client/Server-Prinzip* erfolgt [SE86]. Ausgehend von dem Prinzip der semantischen Dekomposition, also der Zerlegung einzelner DB-Operationen in logisch unabhängige Teilschritte, haben wir ein Client/Server-Modell entwickelt, das die parallele Verarbeitung innerhalb des Anwendungssystems ermöglicht [HKS91, HSS88, Schö90]. Hierbei wurden die folgenden allgemeinen Anforderungen an ein solches Client/Server-Modell zugrundegelegt:

- Jede Komponente kann die ***Rolle eines Servers*** für eine übergeordnete Komponente übernehmen (Auftragnehmer) und ***gleichzeitig als Client*** (Auftraggeber) für weitere Komponenten auftreten. Ein Server erbringt also eine bestimmte Dienstleistung, indem er eine Reihe von Funktionen anbietet, die von den auftraggebenden Komponenten in Anspruch genommen werden können und er kann umgekehrt selbst wieder Dienste von anderen Komponenten anfordern.
- Ein ***Aufruf an einen Server*** (also ein Auftrag) erfolgt ***asynchron***, d.h., er besteht aus der Auftragserteilung und einer davon getrennten Ergebnisentgegennahme. Dies ermöglicht die Erteilung paralleler Aufträge an andere Anwendungskomponenten sowie eine mögliche Weiterverarbeitung in der erteilenden Komponente bis zur nächsten Interaktion mit einem der aufgerufenen Server (etwa Annahme eines Resultats, eines Teilresultats etc.).
- Um mehrere unabhängige Aufträge bzw. rekursiv an sich selbst gerichtete Aufträge absetzen zu können, muß ein Server in der Lage sein, eine ***beliebige Anzahl von Aufträgen entgegenzunehmen*** und diese unabhängig voneinander entsprechend einer vorgegebenen Prioritätenregelung zu bearbeiten.

Entsprechend dem hier vorgestellten Modell erfolgt demnach die Abarbeitung einer Benutzeroperation in einem komplexen Anwendungssystem über eine Zerlegung der Operation in hierarchisch geschachtelte Aufträge und deren parallele Abarbeitung durch die einzelnen Anwendungskomponenten (Server).

2.2 Das RC-System

Nachdem bislang ein erster Anforderungskatalog an ein abstraktes Client/Server-Modell erstellt wurde, ist nun die Frage nach der Bereitstellung und Realisierung eines Basisdienstes zu beantworten, an dessen Schnittstelle die Funktionalität des Client/Server-Modells angeboten wird. Dazu haben wir das ***Remote-Cooperation-System (RC-System)*** entwickelt [HKS91], das die Funktionalität bereitstellt und an seiner Schnittstelle von Betriebssystemeigenschaften der zugrundeliegenden Rechner sowie der aktuellen (möglicherweise sehr heterogenen) Hardware-Umgebung abstrahiert.

Einbettung eines Anwendungssystems in eine Betriebssystemumgebung unter Verwendung des RC-Systems

Die Fragen der Einbettung eines auf dem RC-System basierenden Anwendungssystems in eine Betriebssystemumgebung sind sehr komplex, so daß an dieser Stelle nur die wichtigsten Aspekte aufgezählt werden können. Eine ausführliche Diskussion findet sich in [HKS91, Schö90].

- Als betriebssystemseitige Ablaufeinheit zur Einbettung eines Servers wird das Prozeßkonzept genutzt, das sich insbesondere durch seine generelle Verfügbarkeit sowie die relativ einfache und flexible Verteilbarkeit auf unterschiedliche Rechner anbietet. Für die Zuordnung Server - Prozeß gilt, daß einem Server mehrere Prozesse, einem Server ein Prozeß bzw. mehrere Server einem Prozeß zugeteilt werden können. Das RC-System erlaubt alle drei Möglichkeiten.
- Die Forderung nach einer unabhängigen, nebenläufigen Bearbeitung mehrerer Aufträge (z.B. rekursive Selbstaufrufe) durch einen Server kann durch das Konzept des Multi-Processing bzw. Multi-Tasking sowie durch eine Kombination der beiden Konzepte gelöst werden (zur Begriffswahl vgl. [Me88]). Eine in [Schö90] durchgeführte Bewertung dieser Konzepte macht deutlich, daß das Single-Processing/Multi-Tasking-Konzept zur Abbildung eines Servers in unserer Umgebung ein geeignetes Konzept darstellt, d.h., durch einen Prozeß werden mehrere Aufträge überlappend abgearbeitet.
- Aus allgemeinen Effizienzüberlegungen heraus besitzt das RC-System selbst eine verteilte Systemstruktur. Es gibt also keine zentrale Instanz des RC-Systems, vielmehr sind die Verwaltungsdaten auf die Server-Prozesse verteilt und werden zusammen mit den RC-Systemfunktionen als Unterprogramm an die Systemkomponenten gebunden (vgl. Bild 1).

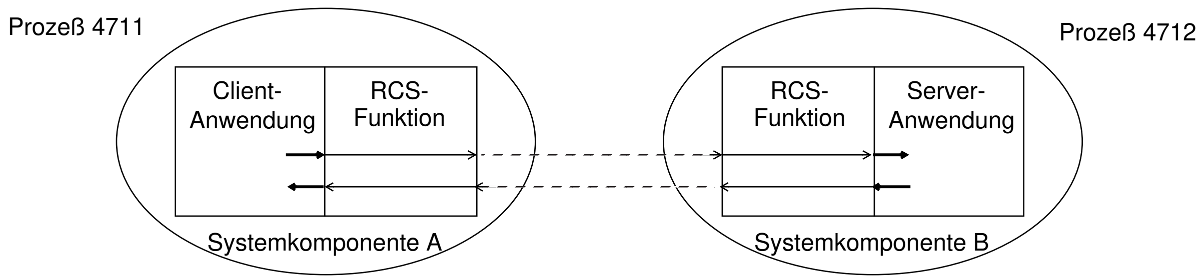

Bild 1: Abbildung der Systemkomponenten auf allgemeine Betriebssystemkonzepte

Abbildung der Server-Prozesse eines Anwendungssystems auf eine heterogene Hardware-Umgebung

Ein weiterer Fragenkomplex richtet sich an die Abbildung eines komplexen Anwendungssystems auf die Hardware-Umgebung. Die Hardware-Umgebung kann durch unterschiedliche Architekturtypen bestimmt sein [HSS89]: ***eng gekoppelte Systeme*** (gemeinsamer Speicher zwischen allen Prozessoren sowie nur eine Betriebssystemkopie, Prozeßverwaltungsschlange etc.), ***lose gekoppelte Systeme*** (kein gemeinsamer Hauptspeicher, nachrichtenorientierte Kommunikation) sowie ***nahe gekoppelte Systeme*** (sowohl eigener Speicher der Prozessoren als auch gemeinsam genutzter Speicher). Für die Abbildung der Server-Prozesse ergeben sich daraus eine Reihe von Problempunkten:

- Verdecken der aktuellen Hardware-Umgebung
  Da die Schnittstelle des RC-Systems unabhängig von der aktuellen Hardware-Umgebung zu gestalten ist, müssen die Hardware-Charakteristika nach außen verborgen werden. Allerdings müssen die besonderen Hardware-Eigenschaften innerhalb des RC-Systems ausgenutzt werden. Dies gilt insbesondere für die Nutzung von gemeinsamem Speicher.

- Flexible und einfache Konfiguration
  Darunter ist die Verteilung des gesamten Anwendungssystems auf die zur Verfügung stehenden Rechner zu verstehen. Das RC-System muß eine einfache und flexible Konfiguration, d.h. Zuordnung der Prozesse zu den Prozessoren, gewährleisten.

Die wichtigsten Operationen des RC-Systems

Im folgenden wollen wir die grundlegenden Operationen des RC-Systems und deren Zusammenwirken etwas genauer vorstellen, soweit dies für das weitere Verständnis notwendig ist. Eine Client-Komponente kann an einen beliebigen Server einen Auftrag absetzen (***Remote_Server_Initiation***). Die RC-Komponente der Client-Seite übernimmt den Auftrag und sorgt für dessen Weiterleitung an die Server-Seite. Unmittelbar nach dem Weiterleiten erhält die Client-Seite die Kontrolle zurück (*asynchrone Auftragserteilung*). Das Ergebnis eines Auftrags kann zu einem späteren Zeitpunkt über die Operation ***Get_Server_Result*** abgerufen werden. Mit ***Look_For_Server_Termination*** kann eine Client-Komponente den aktuellen Zustand eines Auftrags abfragen bzw. mit der Operation ***Wait_For_Server_Termination*** explizit auf das Ende der Bearbeitung eines Auftrags warten.

Die Server-Komponente kann mit der Operation ***Accept_Task*** einen Auftrag zur Bearbeitung entgegennehmen. Stehen mehrere Aufträge zur Bearbeitung an, so wird der nächste zu bearbeitende Auftrag über eine Prioritätensteuerung ermittelt. Hat ein Auftrag Subaufträge abgesetzt, deren Ergebnisse abgewartet werden sollen, so muß der *Auftrag blockiert* werden, d.h. er wird mittels ***Break_Task*** unterbrochen und ein anderer zur Bearbeitung anstehender *Auftrag wird aktiviert*. Trifft zu dem unterbrochenen Auftrag eine Antwort ein, so wird er wieder in die Reihe der *bereiten Aufträge* eingehängt und kann mit einem ***Accept_Task***-Aufruf erneut aktiviert werden. Die Operation ***Reply_Task*** übermittelt das Ergebnis des Auftrags an die Client-Komponente zurück und beendet den Auftrag.

2.3 Die Client/Server-Architektur des PRIMA-Systems

Das PRIMA-System setzt sich aus dem NDBS-Kern und der auf eine definierte Anwendungsklasse orientierten Anwendungsebene zusammen. Der NDBS-Kern, der an seiner Schnittstelle ein Datenmodell zur Beschreibung und Handhabung komplex strukturierter Objekte anbietet [Mi88], ist in weitere Komponenten aufgeteilt, die in Bild 2 (vereinfacht) dargestellt sind. Das das Datenmodell realisierende Datensystem ist in den Server ***Compiler***, der die Anfrageübersetzung und Code-Erzeugung übernimmt, und den Server ***Dml_Ausführung***, der den vom Compiler erzeugten Code ausführt, aufgeteilt. Beide erteilen Aufträge an den Server ***Metadatenverwaltung*** sowie an sich selbst (beispielsweise wenn in einer SELECT-Anweisung weitere SELECT-Anweisungen enthalten sind). Alle Komponenten nutzen den Server ***Transaktionsverwaltung***. Der Server ***Speicherkomponente***, der selbst die E/A-Komponente benutzt, wird hingegen nur vom Server ***Zugriffskomponente*** genutzt, der wiederum nur von der Dml_Ausführung Aufträge erhält. Die Pfeile in Bild 2 zeigen die Auftraggeber/Auftragnehmer-Relation zwischen den Servern. Dabei werden auch die rekursiven Auftragsbeziehungen deutlich. Eine genauere Beschreibung des PRIMA-Systems [HMMS87] ist im Rahmen dieser Arbeit nicht notwendig.

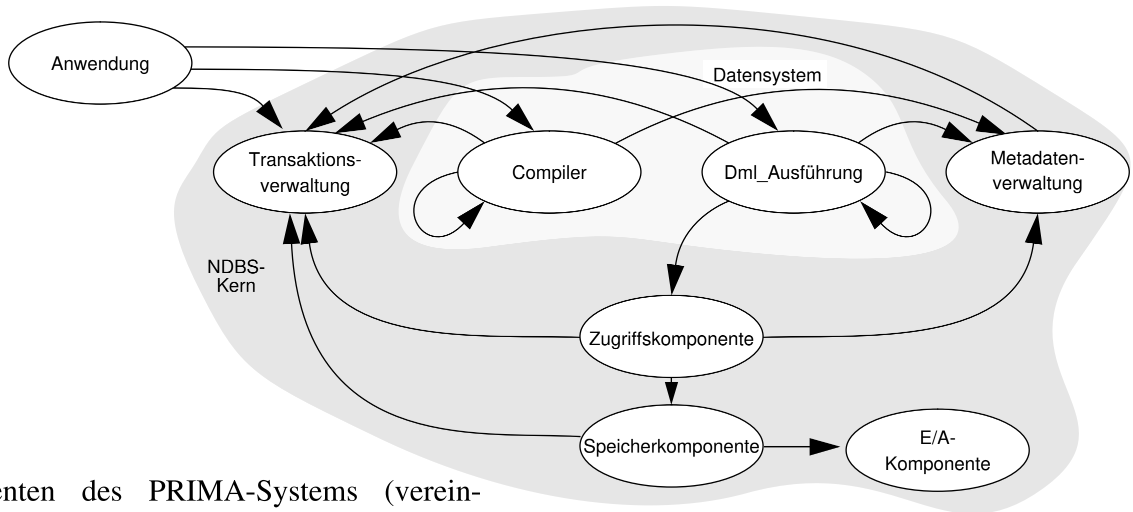

Bild 2:
Server-Komponenten des PRIMA-Systems (verein-

Nachdem wir die Anforderungen an verteilte, parallele Client/Server-Systeme aufgezeigt und deren Realisierung anhand der Funktionalität des RC-Systems geschildert haben, werden im folgenden Kapitel die wichtigsten Systemparameter und Meßgrößen solcher Systeme diskutiert.

3. Systemparameter und Meßgrößen

Das Leistungsverhalten eines parallelen Client/Server-Systems hängt von einer Vielzahl von Parametern ab, die sich grob in drei Klassen einteilen lassen: konfigurationsspezifische Parameter, auftragsspezifische Parameter und implementierungsspezifische Parameter. Zusätzlich zu diesen systeminternen Parametern muß das von außen einwirkende Lastprofil beachtet werden.

Die **konfigurationsspezifischen Parameter** umfassen alle Festlegungen bzgl. der Abbildung des Client/Server-Systems auf eine konkrete Ablaufumgebung. Die unterste Ebene wird dabei durch die vorliegende Hardware gebildet. Hier lassen sich neben den Mono-Prozessoren die drei in Kapitel 2 angesprochenen Architekturtypen unterscheiden. Für das Leistungsverhalten des Systems ist dabei vor allem die Ausnutzung des gemeinsamen Speichers für die Kommunikation bei der Auftragserteilung bzw. der Ergebnisübernahme interessant. Dies gilt besonders, da drei unterschiedliche Typen gemeinsamen Speichers unterschieden werden können:

- gemeinsamer Speicher innerhalb eines Prozesses
  Diese Art des gemeinsamen Speichers ist bei allen Architekturtypen vorhanden und kann zwischen mehreren Servern, die in einem Prozeß ablaufen, genutzt werden.
- gemeinsamer Speicher zwischen Prozessen (eines Prozessors)
  Diese Art des gemeinsamen Speichers wird von fast allen Betriebssystemen angeboten. Allerdings unterscheiden sich sowohl die Zugriffszeiten als auch die Art und Weise des Zugriffs auf diesen Speicher.
- gemeinsamer Speicher zwischen Prozessoren eines Rechners
  - Im Falle der eng gekoppelten Architektur ist nur gemeinsamer Speicher vorhanden, da alle Prozessoren auf den gesamten Speicher zugreifen können (wodurch gleichzeitig die Anzahl der Prozessoren eng begrenzt wird). Gemeinsamer Speicher (in dem hier untersuchten Sinne) entspricht damit dem gemeinsamen Speicher zwischen Prozessen (vgl. vorigen Abschnitt).
  - Im Falle der nah gekoppelten Architektur ist der gemeinsame Speicher zwischen den Prozessoren von dem Arbeitsspeicher der einzelnen Prozessoren verschieden. Dies erlaubt eine deutlich höhere Anzahl von Prozessoren, da sowohl die Zugriffszeit unkritischer ist (die meisten Zugriffe erfolgen auf den privaten Arbeitsspeicher des Prozessors) als auch die Zugriffshäufigkeit auf den gemeinsamen Speicher wesentlich geringer ist (er wird ja nur zur Kommunikation mit anderen Prozessen genutzt). Zusätzlich kann zwischen den Prozessen eines Prozessors gemeinsamer Speicher (im Arbeitsspeicher dieses Prozessors) genutzt werden.

Wegen der flexiblen Konfigurierbarkeit des Gesamtsystems, muß die Nutzung des gemeinsamen Speichers (unabhängig von dessen Typ) vollständig durch das RC-System verborgen werden. Die Ausnutzung des gemeinsamen Speichers ist jedoch als eine wichtige Optimierungsmaßnahme anzusehen.

Die **auftragsspezifischen** Parameter betreffen die Abarbeitungsfolge der Aufträge im System. So sollten z.B. in einem Synchronisationsserver immer die Aufträge zur Freigabe von Sperren vor den Aufträgen zum Anfordern von Sperren bearbeitet werden, da hierdurch eine Verringerung der Blockierungszeiten und des Auftragsaufkommens erreicht wird. Im allgemeinen kann die Abarbeitungsfolge der Aufträge durch den Auftrag selbst, durch den Client und durch den Server bestimmt werden. Zur Steuerung bietet sich eine Prioritätenlösung an, d.h., die Clients und/oder die Aufträge werden mit Prioritäten versehen. Dies erlaubt sowohl die Bevorzugung bestimmter Clients bzw. bestimmter Aufträge als auch eine Kombination von beiden. Die Abarbeitung der Aufträge im Server muß entsprechend dieser Prioritäten erfolgen, wobei weitere Informationen wie beispielsweise die Wartedauer einzelner Aufträge berücksichtigt werden müssen [HKS91], um z.B. ein “Aushungern” von Aufträgen zu vermeiden.

Die **implementierungsspezifischen** Parameter betreffen zunächst die Zerlegung des Anwendungssystems in Server-Komponenten. Da diese Aufteilung in einer sehr frühen Phase der Implementierung erfolgt, ist das resultierende Gesamtsystem in diesem Punkt sehr inflexibel gegenüber Änderungen. Weiterhin spielt die Implementierung der Server, also insbesondere die Auswahl der Algorithmen und Strategien zur Verteilung von Subaufträgen sowie zur Bearbeitung einzelner Aufträge eine wesentliche Rolle. Wiederum ist das Gesamtsystem in diesem Punkt sehr unflexibel gegenüber Änderungen. Die verwendeten Strategien bestimmen darüberhinaus das Auftragsprofil, d.h.

- kurze / lange Aufträge,
- Anzahl der parallel abgesetzten Aufträge,
- Ankunftsrate der Aufträge, etc.

und haben damit wieder Einfluß auf die Systemgestaltung. Neben diesen schwer veränderlichen Parametern, ist noch eine Reihe von Justierungsparametern zu nennen, deren Modifikation durch einfaches Übersezten der Programme möglich ist. Hierzu zählen vor allem Konstanten in den Anwendungsprogrammen und auch im RC-System selbst (beispielsweise die Paketgröße für Nachrichten, Zeiten für Timeouts, etc.), die das Systemverhalten beeinflussen.

Zusätzlich zu den bisher diskutierten systeminternen Parametern muß das von außen einwirkende **Lastprofil** untersucht werden [Bo89]. Hierbei sind insbesondere im Datenbankbereich drei Fragestellungen interessant:

- Referenzverhalten
  Welche Referenzmuster lassen sich beim Zugriff auf die Daten erkennen? (z.B. “80% der Zugriffe betreffen 20% der Daten” oder “Wie häufig wird ein bestimmtes Datum in einer Zeiteinheit referenziert?”)
- Zugriffsverhalten
  Welche Operationen werden mit welchen Häufigkeiten aufgerufen (Zeitverlaufsdiagramm)?
- Konfliktverhalten
  In welchem Maße stehen die ausgeführten Operationen auf den betroffenen Daten im Konflikt zueinander bzw. welches Transaktionsprofil liegt vor?

Die Sensitivität des Systemverhaltens gegenüber einer bestimmten Einstellung der internen Parameter und des von außen einwirkenden Lastprofils spielt bei der Beurteilung und weiteren Untersuchung des Gesamtsystems eine wesentliche Rolle, da bei einer hohen Sensitivität bzgl. des Lastprofils die Wahl der internen Parameter nur im Kontext dieses bestimmten Lastprofils gesehen werden kann.

Die Meßgrößen

Die Analyse und die Bewertung des Systemverhaltens erfordern die Ermittlung aller Meßgrößen, deren Werte durch die Last bzw. durch die Parameterwahl beeinflußt werden. Die Meßgrößen können dabei bzgl. den beiden folgenden Gesichtspunkten betrachtet werden. Zum einen ist die Art der Meßgröße und zum anderen die Systemebene, in der die Meßgröße erfaßt wird, von Interesse. Wie die Betrachtung der einzelnen Aspekte zeigen wird, sind beide Gesichtspunkte nicht immer zu trennen. Im wesentlichen lassen sich sechs Systemebenen in einem parallelen Client/Server-System unterscheiden:

- **Gesamtsystem:** Das Leistungsverhalten aller Prozessoren, Prozesse und Server soll untersucht bzw. bewertet werden.
- **Prozessor:** Alle Prozesse bzw. Server eines Prozessors sollen analysiert werden.
- **Prozeß:** Die Server eines einzelnen Prozesses sollen isoliert untersucht werden.
- **Server:** Das Leistungsverhalten eines einzelnen Servers soll bestimmt werden.
- **Auftrag:** Die Abarbeitung eines Auftrags bzw. seiner Subaufträge stehen im Mittelpunkt der Analyse.
- **Prozedur:** Die Implementierung einer Prozedur soll untersucht werden.

Die Zerlegung des Systems in unterschiedliche Ebenen ist zum Aufbau einer Abstraktionshierarchie notwendig, da nur so die Fülle der Meßdaten auf ein handhabbares Maß reduziert werden kann. Zusätzlich bildet die Abstraktionshierarchie die Grundlage für eine schrittweise verfeinerte Untersuchung des Gesamtsystems. Auf den unterschiedlichen Systemebenen lassen sich die folgenden wichtigen Meßgrößen unterscheiden:

- **Antwortzeit**
  Die (zumindest für den relativen Vergleich) wichtigste Größe zur Beurteilung des Gesamtsystems ist die Antwortzeit für eine Operation bzw. eine Folge von Operationen. Der Vergleich der Antwortzeiten bei gleicher Last mit unterschiedlichen Parameterbelegungen bzw. mit unterschiedlichen Systemkonfigurationen erlaubt Rückschlüsse auf die Sensitivität des Systems bzgl. bestimmter Parameter sowie auf die Eignung der verwendeten Verarbeitungskonzepte. Zum Beispiel ist der Einfluß der Parallelisierung der Verarbeitung beim Vergleich einer Systemkonfiguration mit nur einem einzigen Prozeß (alle Server befinden sich in diesem Prozeß) und einer Systemkonfiguration mit mehreren Prozessen auf einem Prozessor (oder auf mehreren Prozessoren usw.) ableitbar, da bei einer Konfiguration mit allen Servern in einem Prozeß keine echt parallele Verarbeitung möglich ist. Da in diesem Fall die Kommunikation immer über gemeinsamen Speicher erfolgt, sind hierbei auch Hinweise über die Kommunikationskosten bzw. den Einfluß der asynchronen Auftragserteilung ableitbar. Zur Systemanalyse müssen entsprechend die Antwortzeiten für Komponenten des Gesamtsystems gemessen werden.
- **Häufigkeiten**
  Zur Beurteilung der Zerlegung der Operationen und deren paralleler Abarbeitung müssen die Aufrufhäufigkeiten einzelner Operationen bzw. das Auftreten bestimmter Ereignisse protokolliert werden. Dabei muß eine kontextabhängige Aggregierung (z.B. Aufrufhäufigkeiten von Operationen während der Abarbeitung eines bestimmten Auftrags etc.) möglich sein.
- **Auslastung**
  Da alle gemessenen Antwortzeiten von der Auslastung des Gesamtsystems bzw. der entsprechenden Komponenten abhängen, muß die Auslastung jeder dieser Komponenten erfaßt werden. Dies bedeutet, daß die Auslastung der Prozessoren, der Prozesse und der Server bestimmt und im Zusammenhang beurteilt werden muß. Dies ist um so schwieriger, als hier starke gegenseitige Wechselwirkungen auftreten können, beispielsweise kann die Überlastung eines Prozesses die Überlastung eines seiner Server bedingen bzw. die Überlastung eines einzelnen Servers kann zur Überlastung des entsprechenden Prozesses führen.
- **Auftragszustände**
  Die Bearbeitungszustände eines Auftrages (analog zu Prozeßzuständen kann ein Auftrag ***aktiv, bereit*** oder ***blokkiert*** sein) müssen sowohl statistisch erfaßt als auch zeitlich eingeordnet werden können. Als Bearbeitungzustände sind hierbei vor allem das Warten bis zum Beginn der Bearbeitung bzw. das Warten auf die Ergebnisrücknahme sowie alle weiteren Unterbrechungs- und Wartezustände zu nennen. Zusätzlich stellt die Zahl der Zustandsübergänge (und der jeweiligen Verweilzeiten) eine wichtige Information dar.

Bild 3 zeigt den Zusammenhang der Systemebenen mit den Meßgrößen auf. Ein Teil der Meßgrößen ist auf bestimmten Systemebenen nicht erfaß- oder interpretierbar. Zum Teil können jedoch Meßgrößen der darunterliegenden Systemebene für die aktuelle Systemebene aggregiert werden und gewinnen damit eine eigenständige Bedeutung zur Bewertung dieser Systemebene.

	Gesamtsystem	Prozessor	Prozeß	Server	Auftrag	Prozedur
Antwortzeit	o	-	-	o	x	x
Häufigkeiten	o	o	o	x	x	x
Auslastung	o	x	x	x	-	-
Auftragszustände	o	o	o	o	x	-

Legende: x: Messung sinnvoll -: Messung wenig sinnvoll o: Aggregation sinnvoll

Bild 3: Die Bedeutung von Meßgrößen auf den verschiedenen Systemebenen

Zusätzlich zu den in Bild 3 gezeigten Systemebenen können weitere Unterteilungen des Gesamtsystems für dessen Beurteilung hilfreich sein. Hier sind insbesondere Gruppierungen von Systemkomponenten bzgl. bestimmter komplexer Anwendungsfunktionen zu nennen. Ein Mittel zur Gruppierung bildet dabei sicherlich die Auftragshierarchie, die ein wichtiger Indikator für die verwendeten Verarbeitungskonzepte innerhalb des Systems ist. Es muß deshalb möglich sein, die Auftragshierarchie bestimmter Aufträge zu rekonstruieren bzw. statistische Aussagen beispielsweise über die Tiefe und Breite der Auftragshierarchien zu gewinnen. Insbesondere muß feststellbar sein, welche Aufträge innerhalb der Hierarchie parallel bearbeitet werden konnten.

4. Werkzeuge und Bewertungsmethodik

In diesem Kapitel werden zunächst die von uns implementierten Meß- und Analysewerkzeuge vorgestellt und bzgl. den Systemebenen und den Meßgrößen eingeordnet. Im Anschluß wird ihr Einsatz bei der Analyse und Bewertung des Systemverhaltens anhand eines methodischen Vorgehens diskutiert und am Beispiel des PRIMA-Systems verdeutlicht.

Zur Analyse des Leistungsverhaltens unseres Anwendungssystems wurden drei voneinander unabhängige Meßwerkzeuge entwickelt. Dies war nötig, da die von den Betriebssystemen zur Verfügung gestellten Werkzeuge keine gezielte Beobachtung von Servern und Aufträgen, insbesondere des Zusammenhangs der Aufträge über mehrere Server, Prozesse und Prozessoren hinweg, erlauben [Be90]. Die von Betriebssystemen oftmals zur Verfügung gestellten Werkzeuge können allerdings auf innerhalb eines Prozesses Prozedurebene eingesetzt werden [Kl85]. Schritthaltend mit der Erstellung der Meßwerkzeuge wurden Auswertungsprogramme entwickelt, die sowohl eine Beobachtung bzw. eine Analyse des laufenden Systems als auch eine nachfolgende Interpretation erlauben und damit eine möglichst umfassende Untersuchung des Systemverhaltens unterstützen. Die ermittelten Informationen dienen damit sowohl der Veranschaulichung des dynamischen Verhaltens einzelner Komponenten sowie des Gesamtsystems als auch deren Optimierung.

4.1 Meß- und Analysewerkzeuge

Meßwerkzeuge lassen sich aufgrund ihrer Meßmethode in zwei Klassen unterteilen. *Ereignisgesteuerte* Meßwerkzeuge erfassen alle “interessanten” Zustandsveränderungen des Systems. Hierzu müssen diese Zustandsänderungen durch sogenannte “Ereignisse” charakterisiert werden. So wird z.B. die Aktivierung eines Auftrags durch einen bestimmten Aufruf im RC-System veranlaßt. Dieser Aufruf stellt dann das interessierende Ereignis dar. Voraussetzung für diese Meßmethode ist zum einen, daß die Menge der zu messenden Ereignisse ohne spürbare Systembeeinflussung verarbeitbar ist (daß z.B. nicht zu viele Ereignisse in einer Zeiteinheit auftreten), zum anderen, daß man die Ereignisse überhaupt beobachten kann. Aufrufe in der von uns erstellten Software (inkl. RC-System) lassen sich durch Modifikationen der Software protokollieren. Ereignisse im Betriebssystem können wir auf diese Weise jedoch nicht erfassen, denn dazu müßte der Betriebssystemkern modifiziert werden.

In diesem Fall müssen Meßwerkzeuge, die nach der *Sampling*-Methode arbeiten, eingesetzt werden. Diese Meßwerkzeuge erfassen den Systemzustand nur in bestimmten (meist periodischen) Zeitabständen. Wir nutzen diese Methode zur Bestimmung der Prozeßzustände der uns interessierenden Prozesse. Dazu werden die Prozeßzustände in periodischen Zeitabständen abgefragt und entsprechend protokolliert.

Die in Kapitel 3 vorgestellten Meßgrößen werden mittels der drei von uns entwickelten Werkzeuge vollständig erfaßt. Die Einteilung der Meßgrößen in die verschiedenen Systemebenen diente dabei als Grundlage für die Entwicklung der Werkzeuge. Auf der *Prozedurebene* werden durch einen Monitor Prozedurbeginn und -ende protokolliert, um Aussagen über deren Laufzeiten und Aufrufhäufigkeiten zu bekommen. Auf der *Auftragsebene* werden die Basisprimitive des RC-Systems aufgezeichnet, um die logischen Abläufe innerhalb dieser Ebene rekonstruieren zu können und die Aufrufhäufigkeiten von Funktionen der einzelnen Server sowie die Belastung der Server mit Aufträgen aggregieren zu können. Auf der *Serverebene* werden die Auslastung mit Aufträgen und die Aufrufhäufigkeiten von Funktionen beobachtet. Auf der *Prozeßebene* wird von diesen logischen Abläufen abstrahiert. Auf dieser Ebene wird nur noch beobachtet, wieviel, aber nicht was ein Prozeß arbeitet. Im folgenden werden die einzelnen Meßwerkzeuge vorgestellt und die existierenden Auswertungsprogramme skizziert.

Auftrags- und Server-Monitor (ASM)

Mit dem Auftrags- und Server-Monitor werden detaillierte Informationen über die Ausführung eines Programms gewonnen. Die hier ermittelten Daten erlauben das **dynamische Verhalten bei der Auftragsbearbeitung und Auftragserzeugung** sowohl quantitativ (Belastung einzelner Komponenten mit Aufträgen) als auch qualitativ (gezielte Untersuchung der Vorgänge bei der Bearbeitung einzelner Aufträge) zu untersuchen. Im Vordergrund stehen die Untersuchung der Verteilung von Aufträgen zwischen Servern, die Bewertung der Algorithmen innerhalb der Server-Komponenten sowie die Analyse der Laufzeiten bei der Abarbeitung der Aufträge.

Dazu wurde ein *ereignisgesteuerter Softwaremonitor* entwickelt, der die gemäß Bild 3 sinnvollen Meßgrößen auf den Systemebenen Auftrag und Prozedur erfaßt. Dieser Monitor wird in jeden Prozeß unseres Anwendungssystems integriert. Bei Auftreten eines definierten Ereignisses wird ein entsprechender Logsatz in einen Puffer geschrieben, der bei der Initialisierung des Prozesses angelegt wurde. Der gefüllte Puffer wird auf einer lokalen Platte in eine dem Prozeß zugeordnete Datei geschrieben. Da bei einer Messung in der Regel nicht alle möglichen Ereignisse von Interesse sind, kann durch Angabe eines Protokollierungsmodus eine Teilmenge der möglichen Ereignisse zur Protokollierung ausgewählt werden.

Jeder Logsatz besteht aus einem *Header*, der den Zeitpunkt des Ereignisses, den Typ des Ereignisses und eine eindeutige Identifizierung des aktuellen Auftrags enthält. Zusätzlich werden ereignisspezifische *Daten* (z.B. Identifikator eines erzeugten Subauftrags oder Name einer aufgerufenen/beendeten Prozedur) protokolliert. Damit können Aufrufhäufigkeiten, Ausführungsdauer und Aufrufhierarchien von Prozeduren bestimmt werden, indem Prozedurbeginn und -ende zusammen mit dem Prozedurnamen aufgezeichnet werden. Die Abarbeitung eines von einem Client gestellten Auftrags und die dabei entstehende Auftragshierarchie wird durch die Aufzeichnung der Basisprimitive zur Erzeugung von Unteraufträgen (***Remote_Server_Initiation***), zur Unterbrechung eines Auftrags (***Break_Task***) und zur Beendigung eines Auftrags (***Reply_Task***) erreicht.

Eine detailliertere Beobachtung einzelner Aufträge verlangt die Aufzeichnung aller Aufrufe der vom RC-System zur Verfügung gestellten Basisoperationen. Die so erhaltenen Meßprotokolle bilden die Grundlage für eine genauere Untersuchung der implementierten Algorithmen. Die von verschiedenen Prozessoren stammenden Ereignisprotokolle werden dazu zunächst zusammengefaßt. Im Rahmen der durch die Uhren und die Übertragungszeit von Nachrichten vorgegebenen Meßgenauigkeit werden auf allen Prozessoren gleichzeitig der Start-und Endezeitpunkt einer Messung definiert. Die für unsere Meßgrößen benötigten Beobachtungsdauern sind relativ groß, so daß die üblichen Zeitgranulate bzw. Taktzeigen der Prozessoruhren ausreichend genau sind. Unter der Voraussetzung, daß die Uhr auf jedem einzelnen Prozessor gleichmäßig läuft, läßt sich für die Messung eine virtuelle Zeit definieren, die zur Ordnung der Ereignisse von verschiedenen Prozessoren dient. Statistische Auswertungen liefern Aussagen über die Häufigkeiten, mit denen einzelne Funktionen eines Servers aufgerufen werden, über deren Bearbeitungsdauer (getrennt nach den Zeiten in verschiedenen Zuständen) sowie die Erzeugung von Subaufträgen.

Neben den rein statistischen Auswertungen und der Darstellung der Abarbeitungsphasen einzelner Aufträge, gibt der ***Server-Analysator (SA)*** einen visuellen Eindruck von der Lastverteilung im System. Auf einer graphischen Oberfläche werden die Belastungen der einzelnen Server (d.h die Anzahl der zu bearbeitenden Aufträge) sowie das Auftragsaufkommen zwischen einzelnen Servern dargestellt. Dabei werden die einzelnen Server durch Darstellungselemente visualisiert, deren Farben den Belastungsgrad der Server symbolisieren. Variabel breite Verbindungslinien kennzeichnen die Anzahl der Auftraggeber-Auftragnehmer-Beziehungen zwischen den Servern. Durch die Zusammenfassung einzelner Server in einem gemeinsamen Darstellungselement wird eine kompaktere Repräsentation der Server-Belastungen möglich. Dies ist insbesondere dann sinnvoll, wenn mehrere Server in einem Prozeß integriert sind oder das Auftragsaufkommen auf Prozessorebene untersucht werden soll. Bei der Darstellung ist sowohl eine "Zeitraffung" als auch eine Darstellung in "Zeitlupe" möglich.

Online-Server-Monitor (OSM)

Im Gegensatz zur zeitverzögerten Analyse der Ereignis-Protokolle des ASM ist auch eine direkte Beobachtung des Systemverhaltens zur Laufzeit sinnvoll. Zu diesem Zweck wurde ein Monitor entwickelt, der sich unserem Anwendungssystem gegenüber möglichst passiv verhalten sollte und daher seine Meßwerte mittels *Sampling* ermittelt. Im Gegensatz zum ASM werden in diesem Werkzeug die Meßgrößen auf den beiden unteren Systemebenen “Auftrag” und “Prozedur” nicht mehr erfaßt. Der Online-Server-Monitor beobachtet die **Auslastung einzelner Server mit Aufträgen** und die **Häufigkeiten, mit denen einzelne Funktionen eines Servers aufgerufen werden**. Dadurch werden die gewonnenen Informationen zwar unschärfer als im ASM, und die Bearbeitung einzelner Aufträge ist nicht mehr nachvollziehbar, aber die Forderung nach der Passivität wird besser erfüllt. Der OSM kann interaktiv zum laufenden System zugeschaltet werden. Nachdem er mit dem RC-System eines Prozesses Verbindung aufgenommen hat, verschickt dieses in vom Benutzer spezifizierten Intervallen Informationspakete über den aktuellen Zustand der Server des Prozesses. Diese beinhalten die Anzahl der Aufträge, aufgegliedert nach ihren Zuständen und Funktionen, sowie Lauf- und Wartezeiten eines Servers. Für jede Information, die dargestellt werden soll, wie z.B. der Anzahl von Aufträgen, die sich in einem bestimmten Zustand befinden, wird ein entsprechendes Diagramm erzeugt.

Prozeß-Monitor (PM)

Durch die Beobachtung einzelner Prozessoren und Prozesse sollen Engpässe bzw. freie Ressourcen auf den Systemebenen aufgedeckt werden. Da uns eine direkte Beobachtung der Prozeßzustände nicht möglich ist, wurde ein Software-Monitor entwickelt, der in regelmäßigen Intervallen für alle Prozessoren bzw. Prozesse unseres Anwendungssystems aus den Verwaltungstabellen des Betriebssystems die verbrauchten Ressourcen ausliest und über einen Puffer in eine Datei schreibt. Beobachtet werden die **von einem Prozeß aufgenommene Benutzerzeit, die Systemzeit, das E/A-Verhalten und die Prozeßgröße** im realen und virtuellen Speicher. Eine Zusammenfassung dieser Werte für alle Prozesse auf einem Prozessor zur Beobachtungszeit senkt den Einfluß auf das Systemverhalten, da weniger Schreiboperationen notwendig werden. So erfolgt nur noch eine Beobachtung der Auslastung auf Prozessorebene.

In einer anschließenden Auswertung können die so ermittelten Werte über einer Zeitachse aufgetragen werden. Von besonderem Interesse ist dabei die in den einzelnen Zeitintervallen von einem Prozeß aufgenommene Zeit, um ggf. bei einer gleichzeitigen Auswertung von Meßdaten des ASM feststellen zu können, ob dieser Prozeß aufgrund von Aufträgen, die bearbeitet werden könnten, mehr Ressourcen benötigt.

Die vorgestellten Meßwerkzeuge erfassen somit alle in Bild 3 dargestellten, sinnvollen Meßgrößen. Auf den Systemebenen Prozessor und Prozeß werden die Beobachtungen durch den PM, auf der Serverebene durch den OSM und auf der Auftrags- und der Prozedurebene durch den ASM durchgeführt.

4.2 Methodik zur Bewertung und Optimierung von Client/Server-Systemen

Nachdem im vorangegangenen Abschnitt die von uns entwickelten Meß- und Analysewerkzeuge für die Erfassung der in Kapitel 3 aufgezeigten und als relevant erkannten Meßgrößen vorgestellt wurden, geht es nun um deren methodischen Einsatz zur Bewertung und zur Optimierung eines Client/Server-Systems.

Jede Bewertung setzt die Schaffung von Maßstäben und die Spezifikation von Zielgrößen voraus. Nur so können bestimmte Eigenschaften einer konkreten Systemkonfiguration qualitativ und quantitativ eingeordnet werden. Aufgrund ihrer praktischen Relevanz soll im folgenden die Antwortzeit als die wesentliche Zielgröße gelten. In der durch das Client/Server-Modell bereits vorgegebenen verteilten Ablaufumgebung gilt das besondere Interesse der Fragestellung, wie sich die Verteilung, die damit verbundene Möglichkeit einer parallelen Abwicklung von Aufträgen sowie der Einsatz zusätzlicher Ressourcen (zusätzliche Prozessoren, prozessorübergreifender gemeinsamer Speicher etc.) bzgl. dieser Zielgröße bemerkbar macht. Neben der reinen Bewertung einer Systemkonfiguration stellt sich hier natürlich auch die Frage, wie durch die Analyse der erfaßten Meßdaten Hinweise auf eine mögliche Systemverbesserung gewonnen werden können, d.h., wie "Engpässe" erkannt und ihre Ursachen aufgedeckt werden können.

Ein erster Schritt beim Aufdecken möglicher Engpässe im Ablaufverhalten einer Systemkonfiguration besteht in der ***Beobachtung der Prozessorauslastung*** bzw. der Auslastung der einem Prozessor zugeordneten Systemkomponenten. Dies kann zunächst sehr einfach durch den direkten visuellen Eindruck erfolgen, den der Server-Analysator vermittelt. Eine zweite wesentlich feinere Betrachtungsmöglichkeit ist durch den Prozeß-Monitor gegeben, der die prozeß- bzw. prozessorspezifische Belastung aufzeigt. Sind besonders stark ausgelastete Prozessoren zu erkennen, so müssen die dort lokalisierten Server näher betrachtet werden, wobei insbesondere die ***Auslastung der einzelnen Server*** beachtet werden muß. Dabei geht es darum, deren Anteil an der jeweiligen Prozessorauslastung zu erkennen und Hinweise zu erhalten, welche Server in besonderer Konkurrenz zueinander stehen, also gleichzeitig über sehr viele bereite (potentiell verarbeitbare, aktive) Aufträge verfügen. Es empfiehlt sich, diese Server auf verschiedene Prozessoren zu verteilen, d.h., eine Neustrukturierung und Rekonfiguration des Systems durchzuführen. Das Ziel ist dabei, möglichst diejenigen Server auf einem Prozessor zusammenzufassen, die ein tendenziell komplementäres Lastprofil aufweisen, um somit eine möglichst gleichmäßige Auslastung der beteiligten Prozessoren zu erhalten.

In einem zweiten Schritt der Analyse des Ablaufverhaltens geht es besonders darum, die starke Belastung (Anhäufung von Aufträgen, hoher Anteil an den Prozessoraktivitäten etc.) einzelner ***Server näher zu untersuchen***. Hierbei stellt der Auftrags- und Server-Monitor zusammen mit dem Prozeß-Monitor die entscheidende Informationsgrundlage bereit: Stehen beispielsweise in einem Server generell ***viele kurze Aufträge*** (kurze Bearbeitungszeit) zur Bearbeitung an, so verspricht eine Verdopplung dieses Servers (möglichst auf einem anderen Prozessor) eine entsprechende Entlastung. Voraussetzung dafür ist allerdings, daß in der auftragsgebenden Komponenten eine Lastverteilung auf die verschiedenen Server-Ausprägungen möglich ist. Daneben muß, vor allem bei sehr kurzen Aufträgen, die Granularität der vom Server erbrachten Funktionen berücksichtigt werden. Ist die Dauer der eigentlichen Bearbeitung einzelner Auftragstypen sehr gering im Verhältnis zur gesamten Verweildauer dieser Aufträge innerhalb des Servers und im Verhältnis zu den Bearbeitungszeiten anderer Auftragstypen (evtl. auch in anderen Servern), so kann dies als ein Hinweis auf eine zu feine Auftragsgranularität gewertet werden. In einem solchen Fall ist der entstehende Overhead (Taskverwaltung, Parameterübertragung etc.) bezogen auf die vom Server erbrachte Dienstleistung zu groß, so daß hier sinnvollerweise eine Umstrukturierung der Server-Funktionalität durchzuführen ist.

Treten in einem Server ***markante "Langläufer"*** (Aufträge mit einer großen Verweildauer innerhalb eines Servers) auf, so ist dies ebenfalls ein Grund für eine weitergehende Ursachenforschung. Dies kann ggf. innerhalb einzelner Auftragstypen bzw. einzelner Aufträge bis auf die Prozedurebene hin erfolgen. Hierbei sind die folgenden drei Fälle zu unterschieden:

- die Aufträge sind ***lange blockier***t,
- die Aufträge sind ***lange im Zustand bereit***, kommen allerdings nicht zur Verarbeitung und schließlich,
- die Aufträge sind ***lange aktiv*** und bestimmen letztendlich die Server-Belastung.

Blockierte Aufträge warten auf Ergebnisse von Subaufträgen. Nähere Aufschlüsse können daher nur durch die Untersuchung der vom betrachteten Server ausgehenden *Auftragshierarchie* gewonnen werden. Hier können Engpässe innerhalb einzelner Server auftreten, die zunächst zu beseitigen sind. Andererseits kann die Ursache aber auch in der internen Realisierung bestimmter Funktionen des betrachteten Servers liegen, dann beispielsweise, wenn unnötige sequentielle Abfolgen und synchrone Subaufträge anstelle einer möglichst breiten Parallelisierung programmiert wurden.

Im Fall von **bereiten Aufträgen** ist zunächst zu klären, ob die Langläufer innerhalb eines Servers bestimmten *Auftragstypen* zugeordnet werden können, d.h., ob es Server-Funktionen gibt, die im Verhältnis zu anderen Funktionen nur mit niedriger Priorität bearbeitet werden. Eine solche Situation ist nicht generell ungünstig und kann u.U. gewollt sein. Maßnahmen zu deren Behebung sind daher nur dann notwendig, wenn die in einer Auftragshierarchie übergeordneten Aufträge in ihrer Bearbeitung durch diese lang laufenden Subaufträge behindert werden und dann selbst wiederum zu Langläufern werden. In einem solchen Fall kann die Scheduling-Strategie (nach der die zu bearbeitenden Aufträge ausgewählt werden) des beobachteten Servers zu Gunsten der als lang laufend erkannten Auftragstypen beeinflußt werden.

Werden **aktive Langläufer** beobachtet, so ist dies ein Hinweis auf die *Komplexität dieser durch den Server erbrachten Funktion*. Hier ist zunächst zu prüfen (ggf. durch eine prozedurspezifische Analyse innerhalb der Server-Funktionen), ob durch eine algorithmische Verbesserung bei der Realisierung der Server-Funktionen die Bearbeitungszeiten einzelner Aufträge hinreichend reduziert werden können. Ist die realisierte Funktion genügend komplex, so stellt die weitere algorithmische Zerlegung und die Einführung neuer Server zur Arbeitsteilung und parallelen Auftragsabarbeitung eine geeignete Maßnahme dar, diese Art der aktiven Langläufer zu vermeiden.

Damit haben wir folgende iterativ durchzuführende Bewertungs- und Optimierungsschritte herausgearbeitet:

1. Beobachtung der Prozessorauslastung
2. Bestimmen der Anteile der einzelnen Server an der Prozessorauslastung
3. ggf. Rekonfiguration mit dem Ziel einer gleichmäßigen Prozessorauslastung
4. Beobachtung der Aktivitäten der einzelnen Server:
   - Ermittlung markanter Auftragstypen (kurze Aufträge, lange aktive, lange bereite, lange blockierte Aufträge)
   - Untersuchung der Auftragshierarchie, um Zusammenhänge zwischen auffälligen Server-Aktivitäten zu erkennen
   - Untersuchung der Algorithmen und Abläufe zur Realisierung einzelner Server-Funktionen
5. ggf. Durchführung ausgleichender Maßnahmen (Scheduling-Strategie, Restrukturierung/Reimplementierung einzelner Server bzw. Server-Funktionen)

Die Grundidee dieses Vorgehens liegt im Ausgleich zwischen unterschiedlich belasteten Systemkomponenten bzw. zwischen unterschiedlich aufwendigen Server-Funktionen und Auftragstypen. Die vorgeschlagenen Maßnahmen basieren auf der Heuristik, daß die Antwortzeit eines verteilten Client/Server-Systems durch eine möglichst gleichmäßige und gleichartige Belastung aller Systemkomponenten positiv beeinflußt wird. Dies erscheint zwar plausibel, muß allerdings nicht in allen Fällen zu optimalen Ergebnissen führen. Es bleibt daher festzustellen, daß alle konkreten Optimierungsmaßnahmen für jeden Einzelfall isoliert ausgewählt und ergriffen werden müssen. Dennoch stellen die oben aufgeführten Schritte eine Art Grundmuster dar, um mögliche Engpässe innerhalb eines verteilten Client/Server-Systems zunächst einmal zu lokalisieren und schrittweise zu beheben.

4.3 Ein Anwendungsbeispiel

Anhand eines einfachen Beispiels werden wir die Anwendung einiger Werkzeuge und die Tauglichkeit der Bewertungsmethodik demonstrieren. Weitere (komplexere) Anwendungen sind in [Ge91] enthalten. Das Anwendungsbeispiel ist durch eine sehr kleine Datenbank und eine fest definierte Last von 24 Aufträgen, von denen maximal sechs gleichzeitig ausgeführt werden, bestimmt. Unser Anwendungssystem besteht aus den sechs Servern ***Transaktionsverwaltung***, ***Konsistenzverwaltung***, ***Synchronisationsverwaltung*** (in Bild 2 zu Transaktionsverwaltung zusammengefaßt), ***Dml_Ausführung***, ***Zugriffskomponente*** und ***Metadatenverwaltung***. Jeder Server wird in einem eigenen Prozeß realisiert. Zunächst wird nur ein Prozessor genutzt.

Erwartungsgemäß betrug die *Auslastung des Prozessors* 100%. Erste statistische Auswertungen der Meßwerte des ***ASM*** lieferten eine Systemauslastung von 97%, d.h., der zur Verfügung stehende Prozessor hatte im Durchschnitt in 97% der Beobachtungsdauer aktive Aufträge. Die Prozesse hatten durchschnittlich zu 20% der Beobachtungsdauer aktive Aufträge. Dieses macht deutlich, daß bei sechs Prozessen zeitweise mehrere Aufträge auf dem Prozessor gleichzeitig aktiv sein mußten, da sich bei sechs Prozessen eine Belastung von insgesamt 120% ergibt (Schritt 1 der Bewertungsmethodik).

Im gemäß unserer Bewertungsmethodik zweiten Schritt verdeutlichte eine Untersuchung des *Auftragsaufkommens* auf dem Prozessor, getrennt nach den Zuständen aktiv und bereit, daß zeitweise mehr als drei Aufträge gleichzeitig aktiv waren (Bild 4), und daß ständig mehrere Aufträge im Zustand bereit waren.

Im nächsten Schritt (Schritt 3) sollten durch Hinzunahme eines weiteren Prozessors und eine geeignete Umkonfigurierung Verbesserungen erzielt werden. Um diese Umkonfigurierung nicht zufällig durchführen zu müssen, wurden *die einzelnen Server auf ihre Belastung mit Aufträgen* während der ersten Messung untersucht. Dies machte deutlich, daß der Server ***Dml_Ausführung*** das Systemverhalten wesentlich beeinflußte (Bild 5). Die maximale Anzahl von Aufträgen in diesem Server ist größer als 6, da er an sich selbst Aufträge erteilt.

Zur Verbesserung des Systemverhaltens wurde dieser Server in der zweiten Konfiguration auf einen eigenen Prozessor ausgelagert. Neben einer Verkürzung der Gesamtbearbeitungszeit um 20% sank die Auslastung der Prozessoren auf durchschnittlich 64%. Der erneut durchgeführte erste Bewertungsschritt zeigt, daß in dieser Konfiguration nur noch der Prozessor mit dem Server ***Dml_Ausführung*** (Rechner_2 in Bild 6) voll ausgelastet ist. Aus diesem Grund macht es keinen Sinn, über eine Verteilung der anderen Server auf weitere Prozessoren nachzudenken. Dagegen er-

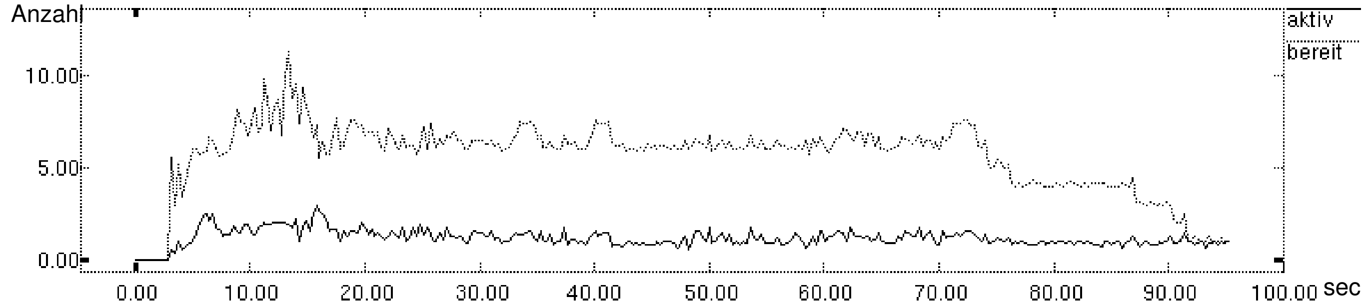

Bild 4: Anzahl der Aufträge auf dem Prozessor in den Zuständen aktiv und bereit über der Beobachtungsdauer

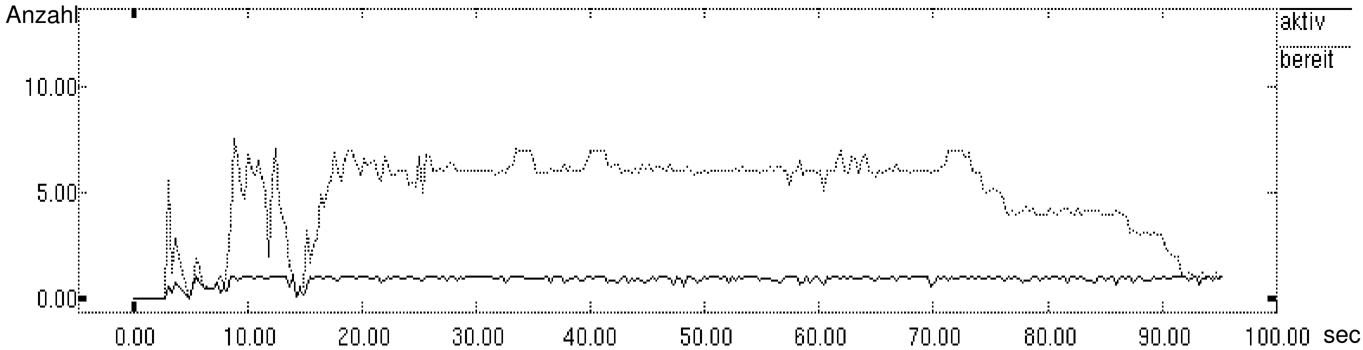

Bild 5: Anzahl der Aufträge im Server **Dml-Ausführung** in den Zuständen aktiv und bereit über der Beobachtungsdauer

scheint eine nähere Betrachtung des Servers ***Dml_Ausführung*** sinnvoll. Die Betrachtung des *Auftragprofils* zeigt, daß dieser Server ständig Aufträge bearbeitet und gleichzeitig Aufträge im Zustand bereit besitzt, also immer noch einen Engpaß darstellt. Die weiteren Maßnahmen können in zweierlei Richtungen gehen: Zum einen kann die *Duplizierung* dieses Servers einen entscheidenden Gewinn bringen (Schritt 3). Wird der neue Serverprozeß auf den ersten Prozessor gebracht, so können durch eine geeignete Lastverteilung beide Prozessoren wieder voll ausgelastet werden. Falls ein weiterer Prozessor zur Verfügung steht, kann dieser für den zusätzlichen Serverprozeß genutzt werden. Zum anderen können in der vierten Phase unserer Bewertungsmethodik die *Verarbeitungsstrategien innerhalb des Servers* genauer untersucht werden. Bild 7 zeigt für einen Auftrag an die ***Dml_Ausführung*** sowie für alle während der Bearbeitung entstehenden Aufträge die Bearbeitungszustände. Zwei Schwachpunkte sind offensichtlich. Der Auftrag mit der Auftragsnummer 1, hier handelt es sich um den Auftrag an die ***Dml_Ausführung***, erteilt in der Regel nur einen Subauftrag und wartet anschließend auf das Ergebnis. Parallelität unter Subaufträgen sowie zwischen Auftraggebern und Auftragnehmern wird also kaum erzeugt. Hier wird also eine Überprüfung der Abarbeitungsstrategie notwendig. Zum anderen ist der Anteil der Wartezeiten bei den Subaufträgen im Vergleich zu den aktiven Zeiten sehr hoch. Diese Wartezeiten wirken sich aufgrund des oben genannten Schwachpunktes direkt auf die Bearbeitungsdauer des Auftraggebers, der in diesen Phasen blockiert ist, aus. In diesem Fall ist zu prüfen, ob eine Änderung der Prioritäten eine Verbesserung erwarten läßt, oder ob die lokale Überlast auf Rechner_1 zu Beginn der Messung (siehe Bild 6) die alleinige Ursache dieser Wartezeiten ist (Schritt 5).

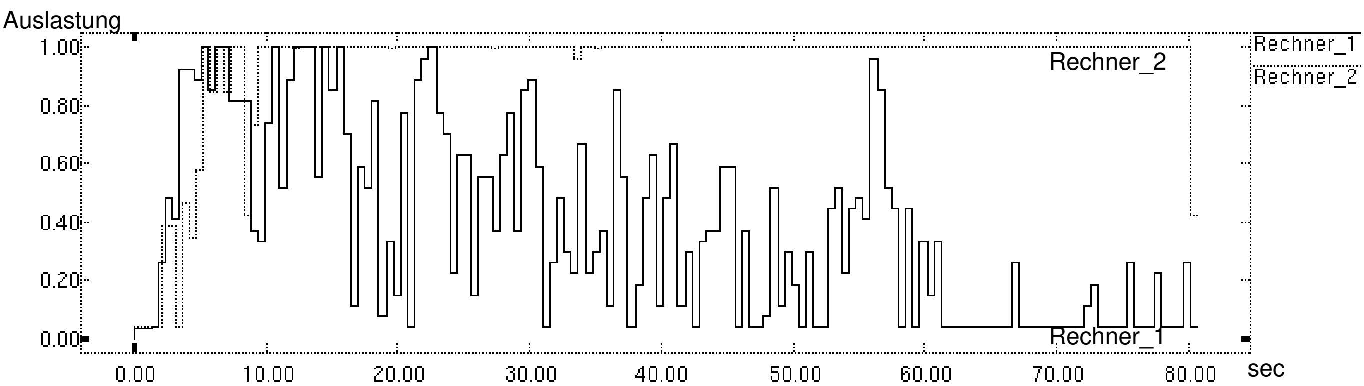

Bild 6: Auslastung der Prozessoren über der Beobachtungsdauer

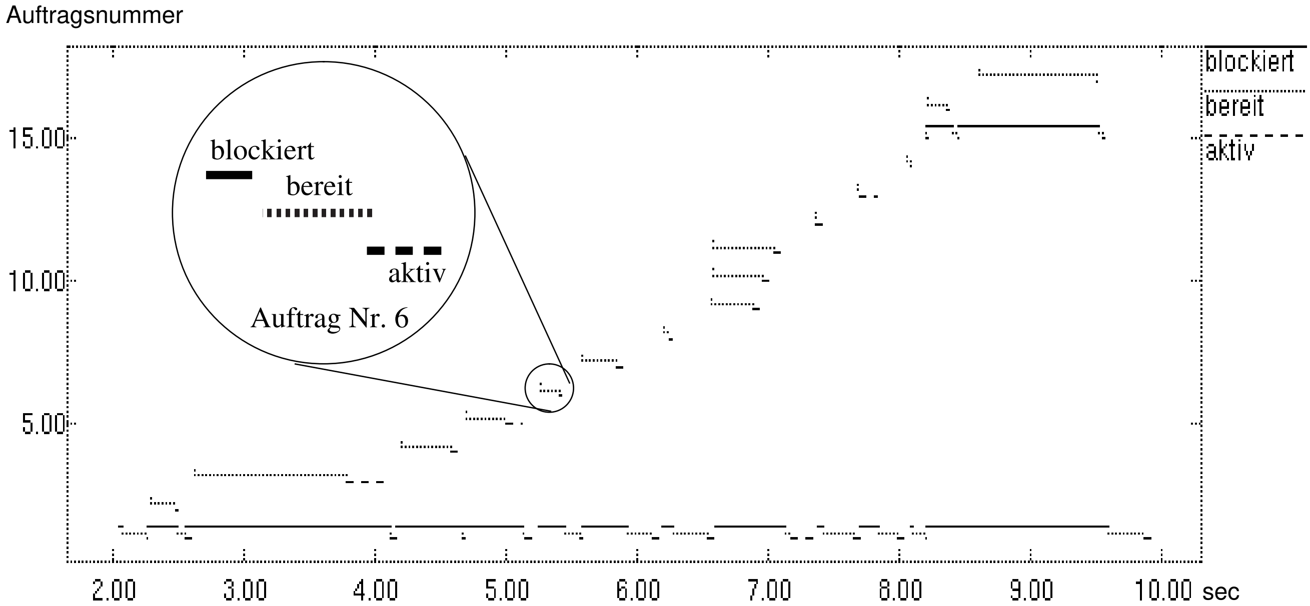

Bild 7: Zustände aller Aufträge bei der Abarbeitung eines Auftrags in der Dml_Ausführung

Mit Hilfe unserer Werkzeuge wurde also in den ersten beiden Schritten unserer in Abschnitt 4.2 vorgestellten Bewertungsmethodik ein Engpaß in der Bearbeitung unserer DB-Last erkannt und reduziert. Nach der Modifikation unserer Konfiguration wurden freie Ressourcen festgestellt, die für weitere Optimierungen durch Konfigurationsänderungen genutzt werden können. Schwachpunkte in der Bearbeitung von Aufträgen wurden erkannt und können nach einer verfeinerten Untersuchungen auf der Auftrags- und Prozedurebene verbessert werden.

5. Zusammenfassung

Am Beispiel des Nicht-Standard-Datenbanksystems PRIMA wurde eine auftragsorientierte Systemmodellierung sowie deren Realisierung in einer konkreten Ablaufumgebung für Client/Server-Systeme vorgestellt. Das im Rahmen des PRIMA-Projektes implementierte RC-System bietet dabei neben einem hohen Maß an Portabilität und Flexibilität bzgl. der zugrundeliegenden Betriebssystem- und Hardware-Umgebung die Basis für die verteilte und parallele Abwicklung von Aufträgen. Die Frage nach einer qualitativen und vor allem nach einer quantitativen Bewertung von verteilten auftragsorientierten Anwendungssystemen wurde ausführlich diskutiert. Hierzu wurden die möglichen Systemparameter und die relevanten Meßgrößen aufgezeigt, klassifiziert und ihre gegenseitige Abhängigkeiten erläutert. Daneben wurde ein Spektrum von speziell entwickelten Meß- und Analysewerkzeugen vorgestellt und ihre Einsatzmöglichkeiten im Rahmen einer methodischen Systembewertung skizziert. Am Beispiel des PRIMA-Systems konnte dieses Bewertungsvorgehen in einer Fallstudie verdeutlicht werden.

Obwohl das RC-System und die damit zusammenhängenden Meß- und Analysewerkzeuge im Rahmen des PRIMA-Projekts entwickelt wurden, stellt es eine allgemeine Ablaufumgebung für verteilte, parallele Client/Server-Systeme dar. Entsprechend kann auch das beschriebene Bewertungsvorgehen bei beliebigen verteilten und auftragsorientierten Anwendungssystemen eingesetzt werden.

Bei der Realisierung der Meßwerkzeuge wurde darauf geachtet, daß ihr Einfluß auf das Verhalten des Gesamtsystems möglich gering ist. Dies geschah eher nach intuitiven Gesichtspunkten, als daß es durch Messungen evaluiert wurde. Für erste Grobanalysen des Gesamtsystems war dies durchaus ausreichend. Allerdings steigt mit zunehmendem Detaillierungsgrad der Analysen der Bedarf einer quantitativen Abschätzung des Einflusses, der durch die Meßwerkzeuge verursacht wird.

Unser weiteres Vorgehen sieht neben einer allgemeinen Analyse weiterer Komponenten sowie des PRIMA-Gesamtsystems eine spezielle Betrachtung einzelner Systemparameter und Einflußfaktoren vor. Insbesondere halten wir eine eingehende Untersuchung der Nützlichkeit von gemeinsamem Speicher in nah oder eng gekoppelten Mehrprozessoranlagen für parallele Client/Server-Systeme für notwendig, um klare, quantifizierbare Aussagen über deren Einfluß auf das Leistungsverhalten des Gesamtsystems zu erhalten.

6. Literatur

Be90 Bemmerl, Th: Integration von Multiprozessorarchitektur, Systemsoftware und Werkzeugumgebung, in: Proc. der GI/ITG-Fachtagung "Architektur von Rechensystemen”, München, 1990.

Bo89 Bohn, V.: Charakteristische Eigenschaften von Transaktionslasten in DB/DC-Systemen, in: Proc. der GI/NTG-Fachtagung "Messen, Modellierung und Bewertung von Rechensystemen und Netzen", MMB-89, Braunschweig, Sept. 1991.

Br88 Brumfield, J.-A., et al: Performance Modeling of Distributed Object-Oriented Database Systems, in: Proc. of the Int. Symp. on Databases in Parallel and Distributed Computing, Austin, Texas, 1988.

Ge91 Gesmann, M.: Untersuchungen zur Parallelität in PHOENIX, Diplomarbeit, Fachbereich Informatik, AG DVS, Universität Kaiserslautern, 1991

HKS91 Hübel, Ch., Käfer, W., Sutter, B.: Ein Client/Server-System als Basiskomponente für ein kooperierendes Datenbanksystem, in: Proc. der GI/ITG-Fachtagung "Kommunikation in Verteilten Systemen", KIVS-91, Mannheim, Feb. 1991.

HMMS87 Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - A DBMS Prototype Supporting Engineering Applications, in: Proc. of the 13th VLDB, Brighton, 1987.

HSS88 Härder, T., Schöning, H., Sikeler, A.: Parallelism in Processing Queries on Complex Objects, in: Proc. of the International Symposium on Databases in Parallel and Distributed Systems, Austin, Texas, 1988.

HSS89 Härder, T., Schöning, H., Sikeler, A.: Evaluation of Hardware Architectures for Parallel Execution of Complex Database Operations, in: Proc. 3rd Annual Parallel Processing Symposium Fullerton, CA, USA 1989, pp 564-578.

KL85 Klar, R.: Hardware/Software-Monitoring, Informatik Spektrum, Bd. 8, Heft 1, Feb. 1985, Springer-Verlag, 1985.

Me88 Meyer-Wegener, K.: Transaktionssysteme, Leitfäden der angewandten Informatik, B.G. Teubner, Stuttgart, 1988.

Mi88 Mitschang, B.: Ein Molekül-Atom-Datenmodell für Non-Standard-Anwendungen, IFB 185, Springer-Verlag, 1988.

Mi90 Miller, B. P., et al: IPS-2: The Second Generation of a Parallel Program Measurement System, in: IEEE Transactions on Parallel and Distributed Systems, Vol. 1, No. 2, April 1990, pp 206-217.

Schö90 Schöning, H.: Realisierungskonzepte für die parallele Bearbeitung von Anfragen auf komplexen Objekten, in: Informatik Fachberichte 266, Springer-Verlag, 1990.

SE86 Seifert, M., Eberle, H.: Remote Service Call: A Network Operating System Kernel for Heterogeneous Distributed Systems, in: NTG-Fachbericht Nr. 92 der GI/NTG Fachtagung “Architektur und Betrieb von Rechensystemen”, Stuttgart, 1986, VDE Verlag.

Zi91 Zimmermann, M.: Configuration Support for Distributed Applications, in: Proc. der GI/ITG-Fachtagung "Kommunikation in Verteilten Systemen", KIVS-91, Mannheim, Feb. 1991.