Claudia Gold

Framework-basierte Unterstützung bei der Realisierung von Lastverteilung

Herbert Utz Verlag · Wissenschaft
München

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Zugleich: Dissertation, München, Techn. Univ., 2003

ISBN 3-8316-0272-7

Printed in Germany

Herbert Utz Verlag GmbH, München
Tel.: 089/277791-00 – Fax: 089/277791-01

Inhaltsverzeichnis

Kapitel 1

Einleitung

Lastverteilung ist seit vielen Jahren Gegenstand der Forschung. Die Beweggründe, die zur Themenstellung dieser Arbeit führten, wollen wir nachfolgend näher betrachten. Im Anschluß geben wir einen Überblick über die einzelnen Kapitel dieser Arbeit.

1.1 Motivation

Mit der Möglichkeit, Programme auf verschiedenen Recheneinheiten verteilt laufen zu lassen, entstand auch der Wunsch, diese möglichst optimal auf die zu Verfügung stehenden Prozessoren zu verteilen und dadurch kurze Programmlaufzeiten und gleichmäßige Ressourcenauslastung zu erzielen. Eine der ersten Arbeiten, die sich mit dem Thema befaßt, liegt gut ein viertel Jahrhundert zurück. 1977 beschreibt Stone [Sto77] ein Verfahren zur optimalen Verteilung der Module eines Programms auf mehrere Prozessoren, bei dem die Ausführungszeiten der Module auf den verschiedenen Prozessoren und das Kommunikationsaufkommen zwischen Modulen berücksichtigt wird. Durch die Verfügbarkeit von Computernetzwerken wie dem ARPANET[1] war zu diesem Zeitpunkt gerade das Interesse am verteilten Rechnen gestiegen. Als Ende der 80er die ersten nebenläufigen Rechensysteme auch für den kommerziellen Einsatz bereit standen, nahm die Bedeutung der Lastverteilung weiter zu. Dies zeigt sich insbesondere durch die hohe Zahl von Veröffentlichungen in dieser Zeit [SHK95]. Die Verknüpfung heterogener Plattformen über Middleware wie PVM [GBD+94] oder die Standardisierung der Interprozeßkommunikation in heterogenen Systemen wie MPI [MPI95] erhöhten den Bedarf an Lastverteilung weiterhin.

Im wesentlichen können zwei verschiedene Arten der Lastverteilung unterschieden werden. Bei der *systembezogenen* Lastverteilung erfolgt die Verteilung von Last ohne Einbeziehung von Anwendungswissen. Damit kommen für die Verteilung nur Einheiten in Frage, auf die von Systemseite zugegriffen werden kann. In den meisten Fällen werden Prozesse als Verteilungseinheiten eingesetzt. Damit zeigt sich auch schon die Problematik der systembezogenen Lastverteilung, die grobe Granularität der Verteilungseinheiten, über die eine gleichmäßige Auslastung der Ressourcen oft nicht erreicht werden kann. Im Gegensatz dazu steht die *anwendungsbezogene* Lastverteilung, bei der Anwendungswissen, wie z.B. die Struktur des Anwendungsproblems, in die Entscheidung einfließen kann. Dies ermöglicht eine Zerlegung des Gesamtproblems in feiner granulare Lasteinheiten und zudem können Abhängigkeiten zwischen den Einheiten, um z.B. den Kommunikationsaufwand zu reduzieren, berücksichtigt werden. Der Gewinn feiner granularen Lastausgleichs kostet den Anwendungsentwickler allerdings zusätzlichen Implementierungsaufwand für

[1] 1969 wurde der erste Hostrechner an das ARPANET angebunden [Lei00]

jedes neue zu realisierende Anwendungsproblem.

Mit zunehmender Bedeutung der Objektorientierung für die Softwareentwickung Anfang der 90er Jahre [Oes97] begann auch das Interesse zu wachsen, Objektorientierung für die verteilte Programmierung einzusetzen. Dies zeigt sich beispielsweise in dem Entwurf der *Object Management Architecture* [OMG95] (OMA) durch die *Object Management Group* (OMG), deren wesentliche Bestandteile schließlich 1995 im CORBA-Standard [OMG99b] spezifiziert wurden. In diese Zeit fällt ebenfalls die Entwicklung der Programmiersprache Java [SUN95], bei der von Anfang an die Verteilung von Objekten ein wichtiges Entwurfskriterium war. So blieb es nicht aus, daß vermehrt auch Objekte für die Verteilung von Last in Betracht gezogen wurden. Für die Bereitstellung systembezogener Lastverteilung auf Objektebene wird diese in die entsprechende Middleware, wie z.B. CORBA oder DCOM [Hor97] integriert. Allerdings ist bislang eine rein systembezogene Lastverteilung nur bei der Verteilung von Dienstanfragen auf die zur Verfügung stehenden Dienstanbieterobjekte anzutreffen. Eine Verlagerung von Objekten zwischen Ressourcen erfordert immer die Einbeziehung von Anwendungswissen, d.h. Wissen über die tatsächliche Objektrealisierung [Lin00, OS01].

Aus oben aufgeführten Einschränkungen systembezogener Lastverteilung folgt unmittelbar, daß sie nicht für alle individuellen Anwendungsprobleme ein adäquates Mittel ist. So wird Lastverteilung häufig direkt vom Anwendungsprogrammierer umgesetzt [Schl97, DSW98, HW98]. Inzwischen wurde eine Reihe von Systemen [ABL+95, GS97, BP97, DHB+00, Dec00, CLZ00] in Form von Sprachmitteln und Programmbibliotheken entwickelt, die dem Anwendungsentwickler bei der Integration von Lastverteilung helfen. Diese Systeme sind häufig auf bestimmte Anwendungsklassen zugeschnitten, unterstützen jeweils nur eine Teilmenge aller möglichen Lastverteilungsmaßnahmen und bieten keine Möglichkeit, die Lastverteilungsstrategie individuell auf das jeweilige Anwendungsproblem und die Systemumgebung abzustimmen. Bietet ein System in Hinblick auf Lastverteilungsstrategien Flexibilität, so wird diese über eine Sammlung von Lastverteilungsstrategien realisiert, aus welcher die am besten geeignete ausgewählt werden kann. Da strategisches Vorgehen jedoch die Einbeziehung von Anwendungs- und Systemspezifika erfordert, können über Strategiesammlungen nur bestimmte Anwendungsklassen effizient unterstützt werden. Alle möglichen Problemklassen können über eine derartige Strategiesammlung nie abgedeckt werden.

Ziel der vorliegenden Arbeit ist deshalb die Unterstützung des Anwendungsentwicklers bei der Integration von Lastverteilung unabhängig von irgendwelchen Anwendungsklassen. Ausgangsbasis ist einzig und allein das objektorientierte Programmiermodell. Da wir auf der Ebene von Objekten arbeiten, bietet sich dabei ein Framework-basierter Ansatz an. Über die Klassen des Framework wird dem Anwendungsentwickler ein Instrument zur Hand gegeben, mit dem er seine Anwendung lastverteilungsgerecht aufbauen kann, so daß die für den Anwendungstyp benötigten Lastverteilungsmaßnahmen genutzt werden können. Neben der Bereitstellung eines Framework zur lastverteilungsgerechten Anwendungsrealisierung ist ein weiterer wichtiger Punkt die Umsetzung geeigneter Lastverteilungsstrategien. Anstelle einer Sammlung von Lastverteilungsstrategien wird dem Anwendungsentwickler ein Baukasten verschiedener Strategiekomponenten bereitgestellt, mit denen er auf einfache Weise eine funktionsfähige Strategie gestalten kann, die den individuellen Anforderungen seiner Anwendung genügt und der jeweiligen Systemtopo-

logie angepaßt ist. Durch eine klare Schnittstellendefinition zwischen Anwendungs- und Strategieseite können Strategien zur Laufzeit ausgetauscht werden. Dadurch wird es dem Anwendungsentwickler ermöglicht, verschiedene Strategien auf ihre Eignung für das jeweilige Problem hin zu testen. Zuletzt sei noch darauf hingewiesen, daß ein solch flexibler Baukasten zur Gestaltung von Strategien eine wertvolle Basis für die Untersuchung von adaptiven Lastverteilungsstrategien, also Strategien, die ihr Verhalten zu Laufzeit ändern, sein kann. Die Adaptivität von Lastverteilungsstrategien war allerdings nicht Thema unserer Betrachtungen, da sie den Rahmen der vorliegenden Arbeit gesprengt hätte.

1.2 Kapitelüberblick

Nachfolgend wollen wir einen Überblick über die weiteren Kapitel der Arbeit geben:

Kapitel 2: Grundlagen
Im Vordergrund dieses Kapitels steht die Einführung in die Thematik der Lastverteilung. Zunächst wird das Lastverteilungsproblem auf Modellebene beschrieben. Um die Integration von Lastverteilung unterstützen zu können, ist das Verständnis des Lastverteilungsregelkreises Grundvoraussetzung. Aus diesem Regelkreis lassen sich die Komponenten herleiten, die sich in den meisten Lastverteilungssystemen wiederfinden. Ferner werden die verschiedenen Mechanismen vorgestellt, die zur Verteilung von Last eingesetzt werden können. Die Unterstützung aller möglichen Mechanismen ist eine wichtige Forderung an ein flexibles und universales Lastverteilungssystem. Die Klassifikation von Lastverteilung dient schließlich der Einordnung der vorliegenden Arbeit. Zudem bildet die vorgestellte Klassifikation von Lastverteilungsstrategien die Ausgangsbasis für den in Kapitel 6 konzipierten Strategiebaukasten. Ein wichtiger Punkt bei der Realisierung von Lastverteilung ist die Gewinnung benötigter Information und deren Austausch, worauf wir in einem eigenen Abschnitt eingehen. Abschließend werden die Konzepte der objektorientierten Softwareentwicklung behandelt, die wesentlich den Entwurf des Lastverteilungssystems bestimmten.

Kapitel 3: Stand der Forschung
Wie bereits erwähnt, ist Lastverteilung seit vielen Jahren ein aktuelles Forschungsthema, was sich auch in der Fülle von Arbeiten in diesem Bereich ausdrückt. Um eine Einordnung der vorliegenden Arbeit zu ermöglichen, werden exemplarisch einige Arbeiten aus den verschiedenen Bereichen der Lastverteilung vorgestellt. Wir beginnen mit Arbeiten zu Lastverteilung auf Prozeßebene und betrachten anschließend die anwendungsbezogene Lastverteilung. Parallelen zur vorliegenden Arbeit findet man besonders in Arbeiten, die Lastverteilung auf Objektebene behandeln und auf die am Ende dieses Kapitels eingegangen wird.

Kapitel 4: Konkretisierung der Themenstellung
Nachdem wir in den vorangegangenen Kapiteln eine Basis für das in dieser Arbeit behandelte Thema *Framework-basierte Unterstützung bei der Realisierung von Lastverteilung* geschaffen haben, wird in diesem Kapitel die Themenstellung näher durchleuchtet. Es werden die einzelnen mit dieser Themenstellung verbundenen Ziele formuliert und die Lösungsansätze zur Erreichung der Ziele skizziert.

Kapitel 5: Das Lastverteilungsframework URAL

In diesem Kapitel wird der Aufbau des Lastverteilungsframework URAL beschrieben. An erster Stelle steht der lastverteilungsgerechte Aufbau des Anwendungsproblems, der durch Bereitstellung eines entsprechenden *Anwendungsrahmens*, konkret durch Vorgabe von Objektklassen, von URAL unterstützt wird. Die Durchführung von Entscheidungen über Lastverteilungsmaßnahmen wird von den Lastverteilungsmechanismen übernommen. Da diese auf Objektebene arbeiten, wird hier Anwendungswissen benötigt. Die Mechanismen werden wiederum als Sammlung von Objektklassen vorgegeben, die vom Anwendungsentwickler entsprechend angepaßt werden müssen. Schließlich werden die Komponenten des Lastverteilungssystems betrachtet, die benötigt werden, um Entscheidungen über Lastverteilungsmaßnahmen zu treffen. Dabei bildet die Einführung der Lastverteilungsdatenbasis eine wichtige Voraussetzung für die Entkopplung von Anwendung und Entscheidungsfindung.

Kapitel 6: Der Strategiebaukasten

Die Entkopplung der Entscheidungsfindung erlaubt eine Trennung der Umsetzung der Lastverteilungsstrategien von der Anwendung und damit auch die Austauschbarkeit von Strategien für eine Anwendung. In diesem Kapitel wird nun ein Konzept vorgestellt, das es ermöglicht, ganz unterschiedliche strategische Vorgehensweisen auf hoher Abstraktionsebene zu spezifizieren und daraus lauffähige Strategien zu erzeugen. Ausgehend von der Analyse von Lastverteilungsstrategien, die sich insbesondere auf eine allgemeine Spezifikation von Strategien stützt, werden verschiedene Bausteine identifiziert, die unterschiedliche, zueinander orthogonale Aspekte einer Strategie umsetzen. Diese können in unterschiedlichen Ausprägungen vorkommen. Die individuellen Strategien ergeben sich dann aus den verschiedenen Kombinationen dieser Ausprägungen. Der Aufbau der einzelnen Bausteine wird mit Hilfe von Klassendiagrammen beschrieben. Anschließend wird die Sprache zur Spezifikation von Lastverteilungsstrategien eingeführt und der Weg von der Spezifikation einer Strategie bis hin zum ausführbaren Lastverteilungsalgorithmus aufgezeigt.

Kapitel 7: Evaluierung anhand von Anwendungsbeispielen

In diesem Kapitel werden die erarbeiteten Konzepte evaluiert. Anhand von drei ganz unterschiedlichen Anwendungsbeispielen wird jeweils gezeigt, wie die Anwendung zu strukturieren ist und wie eine an das jeweilige Problem angepaßte Strategie spezifiziert werden kann. Die Beispiele sind so gewählt, daß Lastverteilungsmechanismen aller Art und ganz unterschiedliche auf die Anwendungsprobleme zugeschnittenen Rechnertopologien zum Einsatz kommen. Durch die Anpaßbarkeit an die verschiedenen Rechnertopologien zeigt sich insbesondere die Flexibilität des entworfenen Lastverteilungssystems in Hinblick auf individuelle Systemgegebenheiten.

Kapitel 8: Zusammenfassung und Ausblick

Abgeschlossen wird die Arbeit mit einer Zusammenfassung der erzielten Ergebnisse und einem Ausblick auf weitere Forschungsarbeiten.