1 Einleitung

> Long before ENIAC was completed, it became clear to the designers that they could utilize the equipment more efficiently if they would adopt serial methods instead of so much parallelism;
>
> *(Knuth [61])*

Die Informatik ist die Wissenschaft der systematischen Speicherung, Verarbeitung und Übertragung von Information. Mit der stetig wachsenden Verfügbarkeit und Produktion neuer Information wachsen in gleichem Maße die Anforderungen an Speicherkapazität, Geschwindigkeit und Komplexität der Verarbeitung, denen sie gegenübersteht. Umgekehrt schafft der Fortschritt auf diesen Gebieten immer neue Einsatzmöglichkeiten der Informationsverarbeitung. Beispiele für herausfordernde Anwendungen in der Industrie sind die Transaktionsverarbeitung von Finanzdienstleistern, das Speichern und Aufbereiten großer Datenmengen in Datenbanken oder die Überwachung und Steuerung komplexer Produktionsprozesse.

Große Herausforderungen finden sich auch auf dem Gebiet der wissenschaftlichen Simulation. Dort sind sie im Englischen als “Grand Challenges” längst zum feststehenden Begriff für wichtige und größte Problemstellungen geworden, die Umfang und Genauigkeit erwarteter Ergebnisse für die mittelbare Zukunft bestimmen. Das “Blue-Book” des Höchstleistungsrechnens der amerikanischen Forschungsförderung [72, 41] definiert

> „A Grand Challenge is a fundamental problem in science and engineering, with broad scientific and economic impact, whose solution can be advanced by high-performance computing.“

Zu diesen Problemen gehören, neben Wirtschaftssimulationen und automatischen Beweisen in der Mathematik, vor allem naturwissenschaftliche Simulationen. Beispiele aus der Physik sind Astronomie und Kosmologie, Strömungsmechanik, Quantenmechanik, Erdmodelle und Klimavorhersage, aus den Ingenieurwissenschaften Simulationen von Fahrzeug-Zusammenstößen oder der Alterung von Atomwaffen und aus der Biologie die Gensequenzanalyse und Berechnung der Proteindynamik.

Die numerische Simulation auf Rechnern, das *wissenschaftliche Rechnen*[1], ist dabei zu einem wichtigen Werkzeug naturwissenschaftlicher Erkenntnis geworden. Die

[1] Im Englischen wird dafür der Oberbegriff „scientific computing“ benutzt und einzelne Disziplinen als „computational physics“, „computational chemistry“ etc. bezeichnet

traditionelle Wissenschaft fußt auf Theorie und Experiment. Sie abstrahiert dabei aus Beobachtungen Modellvorstellungen, die sie in der Theorie mathematisch beschreibt, weiterentwickelt und im Experiment überprüft. Die Komplexität dieser Modelle macht aber häufig ein direktes Verständnis unmöglich. Schon Modelle wie die Navier-Stokes-Gleichungen der Strömungsmechanik sind für die meisten Probleme nicht mehr analytisch lösbar, und Vorgänge auf kleinsten Zeit- und Raumskalen oder in astronomischen Dimensionen sind einem unmittelbaren Experiment nicht zugänglich. Die Visualisierung und die virtuelle Interaktion mit den Simulationsergebnissen erlauben bislang nur schwer erreichbare Einsichten und Erkenntnisse.

Neben den anhaltenden Fortschritten in der Entwicklung sequentieller Rechnerarchitekturen bietet sich zur Bewältigung derart großer Berechnungen das parallele Rechnen als zum herkömmlichen Rechnerentwurf orthogonale Alternative an.[2] Viele Rechner zur Lösung eines Problems zu nutzen, scheint ein kostengünstiger Weg zur Leistungssteigerung über das mit sequentiellen Rechnern Machbare hinaus. Dieser Ansatz ist aber nicht ohne Probleme, sondern stellt neue Herausforderungen dar. Zur Parallelverarbeitung müssen in vielen Gebieten der Informatik, von Hardware und Rechnerarchitektur über Algorithmen, Softwareentwurf und Programmierung neue Konzepte entworfen werden.

Speziell in der Praxis des Softwareentwurfs hat sich die parallele Programmierung trotz einiger erfolgversprechender Ansätze nicht vom zugrundeliegenden Maschinenmodell lösen können, wie das im sequentiellen Rechnen im Beispiel funktionaler oder logischer Programmierung größtenteils möglich war. Neben der verbreiteten Programmierung mit mehreren Kontrollflüssen auf Rechnern mit gemeinsamem Hauptspeicher sind die größten Parallelrechner, die derzeit nahezu 10000 Prozessoren umfassen, mit verteiltem Hauptspeicher konzipiert und werden durch explizite Interprozess-Kommunikation parallel programmiert. Dass dies seit über 10 Jahren gilt, zeigt den Stillstand der praktischen Parallelisierung. Das Diktat der Leistung, das diesem Gebiet seine Existenzberechtigung gibt, verzögert die Annahme neuer Konzepte.

Um so wichtiger ist es, Fortschritte in der Methodik des Softwareentwurfs für das parallele Rechnen nutzbar zu machen. Die wichtigste und einflussreichste Entwicklung war hier die Einführung objektorientierter Methoden. Sie stellt die Strukturierung von Software in Form von Klassen und Objekten ins Zentrum, die, abstrakten Datentypen folgend, im Wesentlichen durch die auf sie anwendbaren Operationen definiert sind. Sie können in Vererbungsbeziehungen stehen (und sind dann Spezia-

[2] Dabei darf die Weiterentwicklung sequentieller Methoden nicht vernachlässigt werden. Ironischerweise kritisiert Gene Amdahl schon 1967 [5] rückblickend: „For a decade prophets have voiced the contention that the organization of a single computer has reached its limits and that truly significant advances can be made only by interconnection of a multiplicity of computers in such a manner as to permit cooperative solution.“ Von wesentlichen praktischen Anwendungen paralleler Architekturen kann aber erst ab etwa 1980 gesprochen werden (siehe z.B. [30]).

lisierungen oder Verallgemeinerung anderer Klassen), und eine Anwendung kann in Form von Abhängigkeiten und Verwendungsbeziehungen zwischen ihnen konstruiert werden. Eine Verbesserung der Qualität von Software scheint dann möglich, wenn durch die Kapselung in Klassen einzeln testbare, korrekte und robuste Komponenten entstehen, die mehrfach wiederverwendbar sind und durch eine Spezialisierung zu späteren Zeitpunkten angepasst werden können. Die Anwendung dieser Konzepte im wissenschaftlichen Rechnen ist damit naheliegend, und eröffnet neue Möglichkeiten und spezielle Fragestellungen.

Motivation und Problemstellung

Diese Arbeit entstand in der Arbeitsgruppe C6 des Sonderforschungsbereichs 382 „Verfahren und Algorithmen zur Simulation physikalischer Prozesse auf Höchstleistungsrechnern“. Dort befassen sich zahlreiche Teilprojekte an den Universitäten Tübingen und Stuttgart mit der Numerik und Physik wissenschaftlicher Simulationen. Als einziges Projekt der Informatik, das an den methodischen Grundlagen der Parallelisierung arbeitet, nimmt die Arbeitsgruppe C6 mit dem Thema „Objektorientierte Parallelisierung“ eine wichtige Rolle ein. Sie versucht objektorientierte Entwurfsmethoden auf wissenschaftliche Simulationen anzuwenden und geeignete Strukturen in Form von Klassen und Entwurfsmustern bis hin zu Klassenbibliotheken zu finden, um beispielsweise Numerik, physikalisches Modell und Parallelität zu entkoppeln oder parallele Algorithmen portabel zu formulieren. Nicht zuletzt sind effiziente objektorientierte Implementierungstechniken, die an Anwendungen exemplarisch vollzogen werden, ein wichtiger Gegenstand der Forschung.

Der letzte Punkt wird in der vorliegenden Arbeit untersucht. Bei der Parallelisierung findet man an der Schnittstelle von Design zur Implementierung einen Bruch. Standards zur parallelen Programmierung sind noch immer an prozeduralen Sprachen und der Kommunikation von einfachen Datenstrukturen orientiert. Das führt dazu, dass objektorientiert entworfene Programme auf Basis dieser Schnittstellen nur unter Umgehung des Objektmodells und der Aufgabe von Typsicherheit parallelisiert werden können, und erhöht damit den Aufwand für Entwicklung und Wartung.

Ziel dieser Arbeit ist es, eine geeignete Grundlage zur Parallelisierung objektorientierter Anwendungen zu schaffen. Dazu ist eine Bibliothek zu entwerfen, die die Kommunikation von Objekten und komplexeren Datenstrukturen in den Mittelpunkt stellt, und so die semantische Lücke zwischen Anwendungsentwicklung und Parallelisierung mit Nachrichtenaustausch schließt. Aus praktischen Gründen sollte eine Lösung kein neues Programmiermodell schaffen, sondern eine geeignete Integration gängiger Konzepte in eine verbreitete Sprache finden. Dabei darf der zentrale Aspekt jeder Parallelisierung, die Leistungssteigerung, nicht aus den Augen verloren werden. Die Möglichkeiten dafür werden in den folgenden Kapiteln erarbeitet, und eine

eigene Lösung wird angegeben.

Die Skizze in Abbildung 1.1 verdeutlicht nochmals graphisch den Bezugsrahmen der vorliegenden Arbeit. Das wissenschaftliche Rechnen bedarf hoher Leistung und komplexer Anwendungen. Diese Anforderungen können durch Parallelisierung und moderne Methoden der Softwareentwicklung erfüllt werden. Die notwendige Brücke zwischen objektorientierter Softwarentwicklung und Parallelisierung schafft diese Arbeit durch das Konzept einer objektorientierten Kommunikationsbibliothek namens TPO++ (Transmittable Parallel Objects in C++).

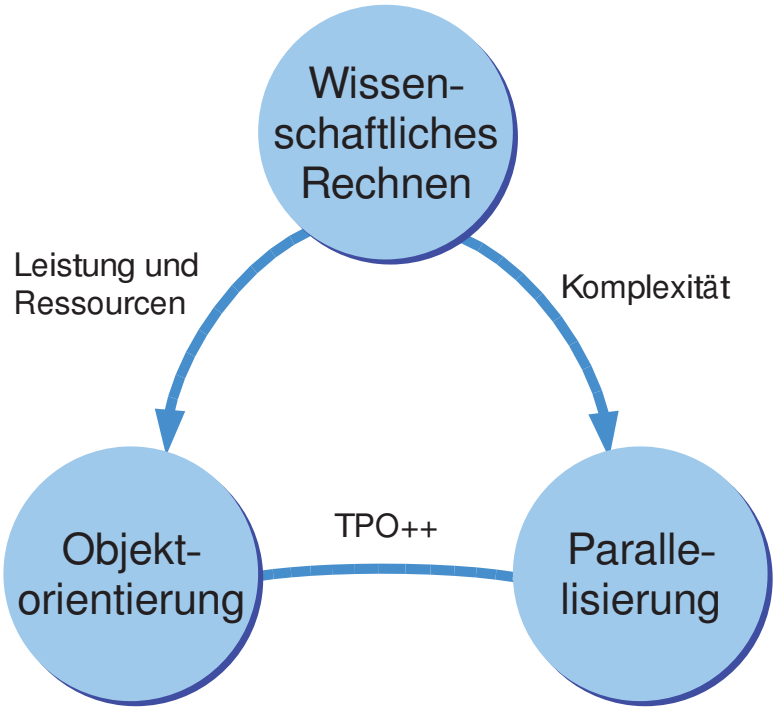

Abbildung 1.1: Bezugsrahmen der objektorientierten Kommunikation mit TPO++ zwischen wissenschaftlichem Rechnen, Objektorientierung und Parallelisierung