

Laufzeitbasierte Entwicklung zweistufig paralleler Programme im wissenschaftlichen Rechnen

Thomas Rauber und Gudula Rünger

Fachbereich Informatik, Universität des Saarlandes, 66041 Saarbrücken

Zusammenfassung: Viele Anwendungen aus dem Bereich des wissenschaftlichen Rechnens weisen zwei Arten von potentielltem Parallelismus auf, Methoden- und Systemparallelismus. Wir stellen ein zweistufiges paralleles Programmiermodell vor, in dem beide Arten durch strukturierte parallele Programme für Maschinen mit verteiltem Speicher in einem Gruppen-SPMD Berechnungsmodell realisiert werden. Die Realisierung besteht aus formalisierten Entscheidungsschritten, deren Güte durch Laufzeitvorhersagen bewertet wird.

1 Entwicklung von parallelen Programmen

Viele potentielle Anwender scheuen vor dem Einsatz von parallelen Systemen zurück, obwohl die Abarbeitung ihrer oft laufzeitintensiven Programme auf herkömmlichen Rechnern zu beträchtlichen Rechenzeiten führt. Der Grund für die Zurückhaltung liegt zum größten Teil an dem erheblichen Aufwand, der notwendig ist, ein sequentielles Programm in ein paralleles Programm umzuwandeln, das die Eigenschaften eines verfügbaren Parallelrechners effizient nutzt. Obwohl mit MPI und HPF mittlerweile Standards für syntaktisch portable parallele Programme existieren, müssen parallele Programme bei der Portierung auf andere Parallelrechner oft umgeschrieben werden, da sich die Effizienz aufgrund der unterschiedlichen Architekturen oft nicht auf einen anderen Rechner übertragen läßt. Eine weitere Problematik der parallelen Programmierung ist es, daß zu Beginn einer parallelen Implementierung nicht klar ist, welche Effizienzeigenschaften das resultierende Programm haben wird. Im schlechtesten Fall stellt sich heraus, daß der verwendete Algorithmus wenig Potential für eine parallele Abarbeitung bietet und das parallele Programm kaum schneller oder sogar langsamer ist als das sequentielle. Hier fehlen geeignete Werkzeuge, die die Laufzeit von Parallelrechnern und verteilten Systemen *vorhersagen*, die also vor der eigentlichen Implementierung eine Abschätzung darüber erlauben, welches Potential an Parallelität ein Algorithmus bietet und welche Laufzeit ein paralleles Programm auf einem speziellen Parallelrechner haben wird.

Das BSP-Modell (*bulk synchronous parallel*) [7] und das darauf aufbauende logP-Modell [1] bieten erste Ansätze, bereits existierende parallele Programme nachträglich zu bewerten. Für die Bewertung eines parallelen Programmes für einen speziellen Parallelrechner mit Hilfe des logP-Modells, werden die Parameter (*l*: *latency*, *o*: *overhead*, *g*: *gap*, *P*: *Anzahl der Prozessoren*) des Modells durch Messungen bestimmt und zur Modellierung des Berechnungs- und Kommunikationsverhaltens verwendet. Da die eigentliche Entwicklung von parallelen Programmen jedoch nicht unterstützt wird, finden diese Modelle in der Praxis wenig Anwendung. Weit in der Praxis verbreitet ist dagegen die Anwendung von klassischen Vektorrechnern, da diese in Form von *vektorisierenden* Übersetzern Hilfestellungen bei der Erstellung von effizienten Implementierungen bieten. Die Entwicklung der entsprechenden Produkte für den Bereich der Parallelrechner, der *parallelisierenden* Übersetzer, steckt dagegen noch in den Kinderschuhen.

In diesem Beitrag stellen wir das integrierte Modell TwoL (*two level*) vor, das eine parallele Programmiermethodik (Auswahlschritte paralleler Programmierentscheidungen) mit einem dazu passenden parallelen Berechnungsmodell (Art der parallelen Abarbeitung), einem Maschinenmodell und einem Analysemodell für Laufzeitvorhersagen verbindet. TwoL erlaubt, im Gegensatz zur weit verbreiteten datenparallelen Abarbeitung, eine Spezifikation zweier übereinander gelagerter potentieller Parallelitätsebenen und deren effiziente Realisierung. Eine ausführliche Beschreibung ist in [6] zu finden.

2 Parallele Programmiermethodik

Jedes Programm ist eine Realisierung von einem oder mehreren Algorithmen, die auf einer hohen Abstraktionsebene beschrieben werden können. Im Falle des wissenschaftlichen Rechnens basieren viele Algorithmen auf numerischen Methoden, die Datenstrukturen wie Vektoren oder Matrizen und entsprechende Operatoren enthalten. Im Gegensatz zu einem Programm in einer üblichen sequentiellen Programmiersprache bietet die abstrakte Beschreibung die Möglichkeit einer Analyse des der Methode innewohnenden Parallelitätsgrades, ohne daß implementierungstechnische Details den Blick auf das Wesentliche erschweren.

Die parallele Programmiermethodik von TwoL erstellt in einem ersten Schritt eine *Modulspezifikation*, die den zugrundeliegenden Algorithmus in geeignete Teilmethoden, sogenannte *Module*, zerlegt. Syntaktische Konstrukte drücken Datenbeziehungen zwischen verschiedenen Modulen aus. Unterschieden wird, ob Module unabhängig voneinander sind und daher parallel zueinander ausgeführt werden können oder ob Datenabhängigkeiten zwischen ihnen bestehen, die eine Reihenfolge der Abarbeitung festlegen. Diese potentielle *Methodenparallelität* stellt die obere Ebene des potentiellen Parallelismus dar. Für viele Programme aus dem Bereich des wissenschaftlichen Rechnens werden innerhalb der Module