Untersuchung von Werkzeugen zur Parallelverarbeitung in Matlab und ähnlichen Systemen hinsichtlich ihrer Eignung für Simulationsanwendungen

R. Fink, S. Pawletta, T. Pawletta
r.fink@et.hs-wismar.de
Hochschule Wismar, FG Computational Engineering und Automation
Philipp-Müller-Straße, PF 1210, D-23952 Wismar

Kurzfassung

Der Beitrag gibt einen Überblick über Werkzeuge zur Unterstützung der Parallelverarbeitung in SCEs[1]. Die Werkzeuge werden anhand ihrer grundlegenden Realisierungsansätze klassifiziert. Für ausgewählte Werkzeuge nach dem Mulit-SCE-Ansatz werden die Ergebnisse einer vergleichenden Leistungsuntersuchung präsentiert.[2]

1 Einleitung

Wissenschaftlich-technische Berechnungsumgebungen (SCEs) wie Matlab, Scilab oder Octave sind seit langem auch als Simulationswerkzeuge etabliert. Bereits die Grundfunktionalität dieser Umgebungen erlaubt die Realisierung komplexer Simulationsanwendungen. Darüber hinaus existieren für einige SCEs spezialisierte Subkomponenten bzw. Toolboxen (z.B. Simulink, Stateflow und Scicos) zur Unterstützung spezifischer Simulationsdomänen.

Simulationsanwendungen, insbesondere wenn es sich um aufwendige Parameterstudien bzw. -optimierungen handelt, erfordern häufig einen sehr hohen Rechenaufwand. Interaktive Entwurfsprozesse werden dadurch stark eingeschränkt oder sogar unmöglich. Dementsprechend enstand in diesem Anwendungsbereich bereits frühzeitig ein Bedarf zur Leistungssteigerung durch Parallelverarbeitung.

Der anfänglich beschrittene Weg der Entwicklung paralleler SCEs führte jedoch nicht zum gewünschen Ergebnis. So berichtete C. Moler in [2] über Experimente mit parallelen Matlab-Versionen, die bereits in den achtziger Jahren durchgeführt wurden und auf Parallelrechnern mit verteiltem als auch gemeinsamem Speicher negative Ergebnisse erbrachten. Zum Zeitpunkt der Veröffentlichung dieser Ergebnisse im Frühjahr 1995 wurde aber bereits an Werkzeugen gearbeitet [3] [4], die die Parallelverarbeitung auf der Ebene von SCE-Anwendungen ermöglichten und mit denen zumindest für grobgranuläre Probleme positive Ergebnisse erzielt wurden.

[1] scientific and technical computing environments

[2] Das Projekt wird in Teilen durch die IAV GmbH und das BM M-V (HWP 310-FHW13) unterstützt.

Die Co-Autoren dieses Beitrags sind seit 1993 auf dem Gebiet der SCE-basierten Parallelverarbeitung aktiv und veröffentlichten 1995 das erste frei verfügbare Werkzeug zur Entwicklung paralleler und verteilter Anwendungen in Matlab (DP-Toolbox [3]). Der zu Grunde liegende und noch heute häufig verwendete Multi-SCE-Ansatz wurde 1998 in [5] beschrieben. Inzwischen wurde eine Vielzahl von Werkzeugen zur Parallelverarbeitung für alle bekannten SCEs entwickelt. Einen vorläufigen Höhepunkt stellt dabei die erste kommerzielle Toolbox für paralleles Rechnen in Matlab dar (DC-Toolbox [7], verfügbar seit Nov. 2004).

Seit Anfang 2004 werden an der Hochschule Wismar im Rahmen eines Forschungsprojekts existierende Parallelverarbeitungsansätze SCE-übergreifend untersucht und gegebenenfalls weiterentwickelt. Dazu wurden in einer ersten Phase mehr als 30 Werkzeuge, haupsächlich nach dem Multi-SCE-Ansatz, identifiziert. Acht Werkzeuge, die verfügbar und lauffähig waren, wurden anschließend analysiert, klassifiziert und einer Leistungsbewertung unterzogen.

Die Ergebnisse der ersten Projektphase werden nachfolgend präsentiert. Dazu werden zunächst die derzeit existierenden grundsätzlichen Ansätze zur Parallelverarbeitung im Zusammenhang mit SCEs aufgezeigt. Die anschließende Ergebnisdarstellung einer vergleichenden Leistungsuntersuchung beschränkt sich auf Werkzeuge basierend auf dem Multi-SCE-Ansatz. Dieser eignet sich strukturell am besten für Cluster- bzw. Gridplattformen und ist damit für einen größeren Anwenderkreis von Interesse. Abschließend werden vier ausgewählte Werkzeuge hinsichtlich ihrer Eignung speziell für Simulationsanwendungen verglichen.

2 Parallelverarbeitung mit SCEs

Die zur Parallelverarbeitung im Zusammenhang mit SCEs verwendeten Ansätze werden in der Literatur unterschiedlich klassifiziert. Die im Folgenden beschriebene Klassifikation stellt eine Zusammenfassung der in [5] und [6] angegebenen Klassifikationen dar:

1. compilerbasierter Ansatz
2. SCEs als Frontend
3. Multi-SCE-Ansatz

Beim compilerbasierten Ansatz wird der sequentielle SCE-Code in ausführbaren Maschinencode übersetzt. Dieser kann entweder selbst parallel ablaufen oder mit parallelen Numerikbibliotheken gekoppelt sein. Da für den Programmablauf keine SCE notwendig ist, dient sie bei diesem Ansatz lediglich als Entwicklungsumgebung.

Der Frontend-basierte Ansatz verwendet eine SCE als Nutzerschnittstelle für parallele Numerikbibliotheken. Der parallele Code wird dabei entweder auf dem lokalen Rechner oder einem entfernten Hochleistungsrechner ausgeführt und bleibt dem Anwender verborgen. Die SCE dient hier als Entwicklungs- und Laufzeitumgebung; für die Ausführung des parallelen Programms ist jedoch nur eine SCE-Instanz notwendig.

Beim Multi-SCE-Ansatz werden mehrere konventionelle SCE-Instanzen unter Verwendung einer Kommunikationsplattform zu einer Multi-SCE verkoppelt. Hier dient der SCE-Verbund wiederum als Entwicklungs- und Laufzeitumgebung.Anders als bei den vorherigen Ansätzen hat der Anwender beim Multi-SCE-Ansatz die volle Kontrolle über den parallelen Code. Da dieser direkt in das ursprünglich sequentielle Programm eingefügt wird, spricht man auch von einer Parallelisierung auf Applikationsebene. Diese Arbeitsweise erfordert zwar einerseits vom Programmierer höhere Sorgfalt und Fachkenntnis, andererseits führt die interaktive Arbeitweise mit SCEs zu schnellen Ergebnissen bei der parallelen Programmentwicklung. Da beim Multi-SCE-Ansatz keine Hochleistungsrechner benötigt werden, eignet er sich besonders für Clustercomputer und Workstation-Netze und wurde im Rahmen des Forschungsprojektes detaillierter betrachtet.

Beispielhafte Wekzeuge, die sich eindeutig einem einzelnen Ansatz zuordnen lassen, sind in Tabelle 1 aufgeführt. Es sei jedoch angemerkt, dass auch Werkzeuge existieren, die auf mehr als einem Ansatz basieren und in diesem Sinne hybrid sind.

Ansatz	Vertreter
compilerbasiert	FALCON, Otter, RTExpress, CONLAB
SCEs als Frontend	NetSolve, DLab, Matpar, MATLAB*P
Multi-SCE-Ansatz	DP-Toolbox, DC-Toolbox, Scilab, Octave-MPITB, Parallel (GAUSS), MPIDL, ...

Tabelle 1: Ansätze zur Parallelverarbeitung mit SCEs und Vertreter

3 Kommunikationsleistung von Multi-SCEs

Da Clustersysteme über eine vergleichsweise geringe Kommunikationsleistung verfügen, war die effiziente Ausnutzung dieser begrenzten Resource bei der Untersuchung von Multi-SCEs von besonderem Interesse. Für die Messung der Kommunikationsleistung wurde das Round-Trip-Verfahren gewählt, bei dem ein Datum zwischen zwei SCE-Instanzen sowohl hin- als auch rückgesendet wird, während eine Laufzeitmessung die Übertragungsdauer bestimmt. Durch Regression über die Übertragungsdauer bei verschiedenen Datenmengen wurden die Leistungsparameter Übertragungsrate und Latenzzeit bestimmt. Die Messungen wurden auf einem Linux-Cluster, bestehend aus 12 AMD-Athlon Prozessoren (1500 MHz), gekoppelt durch ein Gigabit Ethernet, durchgeführt. Tabelle 2 stellt die Ergebnisse unterschiedlicher Multi-SCEs in Zusammenhang mit ihren Kommunikationsplattformen gegenüber.

Die Tabelle zeigt, dass die ermittelten Latenzzeiten teilweise stark differieren. Dabei besitzen Multi-SCEs, die Message-Passing-Systeme wie PVM oder LAM benutzen, Latenzzeiten im Bereich von TCP-Sockets, während die Latenzzeiten anderer Werkzeuge um mehrere Zehnerpotenzen darüber liegen. Die mit Abstand höchste Latenzzeit weist in dieser Untersuchung das einzige kommerzielle Werkzeug (DC-Toolbox von The MathWorks) auf,

Multi-SCE	Kommunikations-plattform	Latenzzeit [ms]	Übertragungs-rate [MB/s]
Referenz-Multi-SCE	TCP-Socket	0.04	37.65
MPITB	LAM (MPI-2)	0.09	36.66
Octave-MPITB	LAM (MPI-2)	0.13	37.18
Scilab	PVM	0.16	11.96
DP-Toolbox	PVM	0.67	11.98
Matlab Parallelization Tk.	Matlab-Engine	25.34	37.16
MatlabMPI	NFS	44.77	25.51
Beolab-Toolbox	Matlab-Engine	81.78	37.09
Matlab DC-Toolbox	unbekannt	1067.48	2.50

Tabelle 2: Kommunikationsleisung von Multi-SCEs

bei dem für eine Datenübertragung mindestens eine Sekunde benötigt wird.

Hinsichtlich der Übertragungsrate sind die Unterschiede zwischen den untersuchten Werkzeugen weniger auffällig, der Abstand zu TCP-Sockets ist hier ebenfalls geringer. Eine Ausnahme stellt wiederum die DC-Toolbox dar, die eine um den Faktor 15 verringerte Übertragungsrate aufweist. Weil keine Informationen über Interna der DC-Toolbox von The MathWorks bereitgestellt werden, konnten die Ursachen für die schlechten Ergebnisse dieses Werkzeugs bisher nicht geklärt werden.

4 Eignung von Multi-SCEs für Simulationsanwendungen

Neben der Kommunikationsleisung ist die Gestaltung der Programmierschnittstelle ein wesentliches Kriterium für die Leistungsbewertung von Multi-SCEs in den jeweiligen Anwendungsbereichen.

Um die Eignung von Multi-SCEs für Simulationsanwendungen bewerten zu können, wurde mit vier ausgewählten Werkzeugen unter drei verschiedenen SCEs das Testproblem *SNE CP1* (Monte-Carlo-Studie) parallelisiert. Gegenüber der ursprünglichen Definition des Testproblems in [1] wurde wegen der inzwischen stark gestiegenen Rechenleistung der verfügbaren Hardwarebasis der Simulationshorizont von 2 auf 10 Sekunden vergrößert. Abbildung 1 zeigt den Verlauf des Speedups sowie die Laufzeit der sequentiellen Programmvarianten (T_seq).

Die Ergebnisse der Laufzeitmessungen zeigen, dass mit allen Werkzeugen eine Beschleunigung durch Parallelisierung erreicht werden konnte. Dabei erzielte die DP-Toolbox Laufzeitgewinne im Bereich des idealen Speedups (Speedup ideal=Anzahl Prozessoren), während die Scilab/PVM-Toolbox nur etwa die Hälfte des idealen Speedups erreichte. Die Ursache dafür ist jedoch nicht die geringere Leistungsfähigkeit des Werkzeugs, sondern die vergleichsweise kurze Laufzeit der sequentiellen Programmvariante. Für einen parallelen Programmablauf müssen die meisten Multi-SCEs (Ausnahme: DC-Toolbox) zunächst

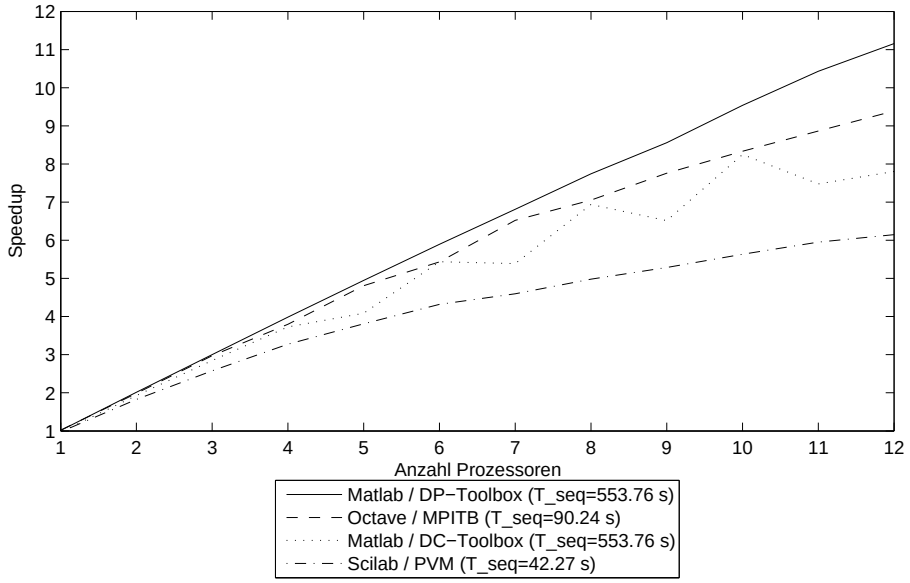

Abbildung 1: Parallele Monte-Carlo-Studie: Laufzeitergebnisse und Speedup

mehrere SCE-Instanzen (Prozesse) starten. Diese Instantiierungsphase stellt einen nicht-parallelisierbaren Programmanteil im Sinne des Ahmdahlschen Gesetzes dar und setzt dem Speedup eine theoretische Obergrenze. Bei relativ kurzen sequentiellen Programmlaufzeiten erhöht sich dieser nicht-parallelisierbare Anteil und führt somit zu geringeren Speedup-Werten.

Um den Entwicklungsaufwand der parallelen Programmvarianten zu quantifizieren, wurden diese hinsichtlich der Anzahl ihrer Programmzeilen untersucht und mit den jeweiligen sequentiellen Programmen verglichen (s. Tab. 3). Das günstigste Zeilenverhältnis zwischen paralleler und sequentieller Programmvariante konnte hier mit der DP-Toolbox unter Verwendung der enthaltenen *Scatter/Gather*-Operationen erreicht werden. Da ähnliche Operationen in den anderen untersuchten Werkzeugen fehlten und somit zusätzlich programmiert werden mussten, verdoppelte sich hier meist die Anzahl der Codezeilen gegenüber den sequentiellen Programmvarianten.

Multi-SCE	Codezeilen sequentiell	Codezeilen parallel	Zeilenverhältnis (parallel/sequentiell)
DP-Toolbox	21	34	1.62
Matlab DC-Toolbox	21	43	2.05
Scilab	24	52	2.17
Octave-MPITB	25	56	2.24

Tabelle 3: Entwicklungsaufwand bei unterschiedlichen Multi-SCEs

5 Zusammenfassung

Die bisherigen Untersuchungen haben gezeigt, dass Simulationsanwendungen mittels Multi-SCEs deutlich beschleunigt werden können. Dabei leistet die interpretative Arbeitsweise von SCEs einen enormen Beitrag zur schnellen Programmentwicklung und Fehlersuche.

Die Untersuchung von Multi-SCEs zeigt jedoch auch, dass die verschiedenen Werkzeuge individuelle Stärken und Schwächen besitzen. So haben Werkzeuge wie z.B. MPITB für Matlab und Octave eine sehr hohe Kommunikationsleistung, verfügen aber häufig nur über eine komplizierte Nutzerschnittstelle auf zu geringem Abstraktionsniveau. Besonders auffallend in dieser Untersuchung ist die schwache Kommunikationsleistung der Matlab-DC-Toolbox, deren Ursache jedoch bisher nicht geklärt werden konnte.

Literatur

[1] *F. Breitenecker, I. Husinsky, G. Schuster:* Comparison of parallel simulation techniques, in EUROSIM-Simulation News Europe, Ausgabe 10, Seite 21f., März 1994.

[2] *C. Moler:* Why there isn't a parallel MATLAB. MATLAB News and Notes, p. 12, Spring 1995.

[3] *S. Pawletta, W. Drewelow, P. Dünow, T. Pawletta, M. Süße:* A MATLAB toolbox for distributed and parallel processing, 2nd International MATLAB Conference, Cambridge, MA, October 1995.

[4] *V. Menon, A. E. Trefethen:* MultiMATLAB: integrating MATLAB with high-performance parallel computing, in Proceedings of the 1997 ACM/IEEE conference on Supercomputing, November 1997.

[5] *S. Pawletta:* Erweiterung eines wissenschaftlich-technischen Berechnungs- und Visualisierungssystems zu einer Entwicklungsumgebung für parallele Applikationen, Dissertation, Universität Rostock, Juni 1998. Erschienen bei ARGESIM/ASIM-Verlag, Wien, 2000.

[6] *R. Choy, A. Edelman:* Parallel MATLAB: Doing it Right, Technical Report, Computer Science AI Laboratory, Massachusetts Institute of Technology, November 2003.

[7] *The MathWorks:* Distributed Computing Toolbox User's Guide, Version 1.0, November 2004