Lösung großer linearer Gleichungssysteme in Linux-Clustern mittels MPI und PETSc

Jens Oeser und Ralph-Uwe Börner[1]

TU Bergakademie Freiberg

1 Einleitung

Die Berechnung der elektrischen Potentialverteilung für dreidimensionale Leitfähigkeitsstrukturen gestaltet sich numerisch extrem aufwendig, wenn Detailaussagen zur Interpretation eine feine Diskretisierung der Parameterverteilung erfordern. Der Rechenaufwand wird im Wesentlichen von der Lösung linearer Gleichungssysteme dominiert. Durch den Einsatz von parallelen Gleichungslösern kann die benötigte Rechenzeit im Vergleich zu Einzelprozessormaschinen gesenkt werden. Auf Grund der hohen Kosten bei der Anschaffung und Unterhaltung von Parallelrechnern werden dabei Cluster aus vernetzten Computern immer interessanter. Im Vordergrund unserer Untersuchung stehen Tests in einem heterogenen Linux-Cluster aus handelsüblichen, durchschnittlich ausgerüsteten PCs, die verbunden sind über ein Netzwerk mit einer Übertragungsrate von 100 Mbit/s (Fast Ethernet). An Hand von Maßzahlen zur Beschreibung der Effizienz paralleler Gleichungslöser wird gezeigt, dass in einem derartigen Netzwerk beachtliche Leistungssteigerungen erzielt werden können.

2 Das Vorwärtsproblem

Viele Fragen der mathematischen Physik führen auf partielle Differentialgleichungen, die mit der Methode der Finiten Differenzen oder Finiten Elemente näherungsweise gelöst werden können. Als Beispiel soll hier das Problem der dreidimensionalen Widerstandsgeoelektrik dienen. Für eine Stromquellenfunktion Q wird bei vorgegebener Leitfähigkeitsverteilung σ die Potentialverteilung φ im Halbraum aus der Diskretisierung der Differentialgleichung

$$\nabla \cdot (\sigma(\vec{r})\nabla\varphi(\vec{r})) = Q, \tag{1}$$

$$Q = -I \cdot \delta(\vec{r} - \vec{r}_s) \tag{2}$$

gewonnen. Q ist der Quellenterm, der für den Fall einer punktförmigen Einspeisung am Ort $\vec{r}_s$ die Form (2) annimmt. Hier bezeichnen I die Stromstärke und δ die Diracsche Deltafunktion.

Die Berechnung der Potentialverteilung erfolgt mit einem 3D-Finite-Differenzen-Algorithmus (Börner et al., 1998). Hier wird neben dem Schema zur Volumendiskretisierung nach Dey & Morrison (1979) eine Technik eingesetzt, die zur Beseitigung der Quellensingularitäten eine Aufspaltung des Potentials in einen normalen und anomalen Anteil verwendet. Damit enthält Gl. (1) modifizierte Einträge im Quellenterm, die durch Leitfähigkeitskontraste und das leicht zu berechnende Primärpotential eines homogenen Halbraums bestimmt sind. Die Approximation dieser modifizierten Gl. (1) über Finite Differenzen führt schließlich auf ein Gleichungssystem der Form

$$\mathbf{A} \cdot \varphi = \mathbf{b} \tag{3}$$

wobei die Koeffizientenmatrix $\mathbf{A}$ dünn besetzt, symmetrisch und positiv definit ist. Die rechte Seite $\mathbf{b}$ enthält die Quellenverteilung. Ein lineares Gleichungssystem der Form (3) kann auf vielfältige

[1] oeser@student.tu-freiberg.de, rub@geophysik.tu-freiberg.de

Weise gelöst werden. Die besonderen Eigenschaften der Matrix $\mathbf{A}$ legen nahe, sogenannte Krylov-Unterraum-Verfahren zu verwenden. Prominentester Vertreter ist die Methode der Konjugierten Gradienten (CG) (Hestenes und Stiefel, 1952). Der wesentliche Vorteil dieser nichtstationären iterativen Methode besteht in der größeren Geschwindigkeit im Vergleich zu direkten Verfahren, wie etwa dem Gaußschen Eliminationsverfahren, sowie stationären iterativen Methoden, z.B. dem Gauß-Seidel-Verfahren. Die Konvergenzgeschwindigkeit der Methode der Konjugierten Gradienten hängt entscheidend vom Eigenwertspektrum und der Konditionszahl der Koeffizientenmatrix $\mathbf{A}$ ab. Eine Konvergenzbeschleunigung erzielt man durch geeignete Vorkonditionierung der Matrix $\mathbf{A}$, indem man das äquivalente Gleichungssystem

$$(\mathbf{M}^{-1}\mathbf{A})\varphi = \mathbf{M}^{-1}\mathbf{b} \tag{4}$$

löst. Es existieren verschiedene Methoden zur Berechnung der Vorkonditionierungsmatrix $\mathbf{M}^{-1}$, deren Eigenschaft darin besteht, die Inverse von $\mathbf{A}$ approximativ, d.h. mit $\mathbf{M}^{-1}\mathbf{A} \approx \mathbf{I}$, darzustellen. $\mathbf{I}$ bezeichnet die Einheitsmatrix. Trotz erhöhtem numerischen Aufwand durch Bereitstellung der Vorkonditionierungsmatrix ist die Konvergenzgeschwindigkeit erheblich größer.

Die rechenzeitbestimmenden Operationen innerhalb eines Iterationsschrittes k mit Konjugierten Gradienten bestehen in der Berechnung von Vektor-Vektor-Produkten der Form

$$\rho_k = \mathbf{r}_k^T \mathbf{r}_k$$

sowie Matrix-Vektor-Produkten mit Vektoraktualisierung der Form

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$$

(Spitzer, 1995). An diesem Punkt können Parallelisierungsstrategien wirksam werden. Der von uns verwendete 3D-Algorithmus bedurfte erheblicher Modifikationen. Die Programmabschnitte, welche die Koeffizientenmatrix und die rechte Seite von (3) erzeugen, sowie die CG-Routine zur Lösung des Gleichungssystems mussten neu geschrieben werden.

3 Verteiltes Rechnen in Netzwerken

Der Rechenzeitbedarf zur Lösung der Gleichungssysteme wächst überlinear mit der Anzahl an Unbekannten. Für große Probleme ist es daher nötig, verschiedene Strategien zur Effizienzsteigerung einzusetzen. Neben der Nutzung schneller Gleichungslöser ist die Idee naheliegend, mehrere vernetzte Rechner einzusetzen, um den Gesamtbedarf an Rechenzeit zu minimieren. Grundlegende Voraussetzung für verteiltes Rechnen in Netzwerken ist eine geeignete Kommunikationsbibliothek. Als Quasi-Standard hat sich das Message Passing Interface (MPI) (Pacheco, 1997) als Umsetzung des nachrichtenorientierten Programmiermodells etabliert. Der durch den Einsatz mehrerer Rechner erzielbare Geschwindigkeitszuwachs hängt von verschiedenen technischen Faktoren, wie Anzahl der Knoten, Netzwerktopologie sowie Datenübertragungsrate ab.

3.1 Leistungsbewertung

Auf den erreichbaren Laufzeitgewinn hat nicht nur die verwendete Hardware, sondern auch in starkem Maße die Art und Weise der Implementierung der numerischen Algorithmen einen Einfluss. Es werden spezielle Kriterien zur Bewertung benutzt. Unter Verwendung von P Prozessoren ist bei vernachlässigbarem Kommunikationsaufwand theoretisch eine Reduzierung der Laufzeit auf $1/P$ möglich. Der Faktor des Laufzeitgewinns wird als Speedup bezeichnet. Für eine feste Problemgröße wird der Speedup berechnet über

$$S(P) = \frac{t(1)}{t(P)}, \tag{5}$$

wobei $S(P)$ den Speedup für P Prozessoren, $t(1)$ die Laufzeit eines Prozesses für einen Prozessor und $t(P)$ die Laufzeit für P Prozessoren bezeichnet. Im Zusammenhang mit dem Speedup steht die Effizienz, die nach

$$E(P) = \frac{t(1)}{P \cdot t(P)} = \frac{S(P)}{P} \tag{6}$$

berechnet wird. Aus der Effizienz lassen sich Aussagen über die Effektivität der Umsetzung eines parallelen Algorithmus im Vergleich zur Einzelprozessormaschine ableiten.

3.2 Parallelisierung

Wie bereits in Abschnitt 2 im Zusammenhang mit der Methode der Konjugierten Gradienten beschrieben wurde, besitzen Vektor-Vektor-Produkte, Matrix-Vektor-Produkte sowie Vektor-Updates ein großes Parallelisierungspotential. Am Beispiel des Matrix-Vektor-Produktes soll gezeigt werden, wie die Verteilung der numerischen Arbeit auf mehrere Prozessoren umgesetzt werden kann. In Abbildung 1 wird die Aufteilung der Zeilen einer Matrix und eines Vektors auf vier Prozessoren skizziert. Es befinden sich jeweils die gleichen Zeilen des Vektors und der Matrix auf einem Prozessor. Zur Berechnung des

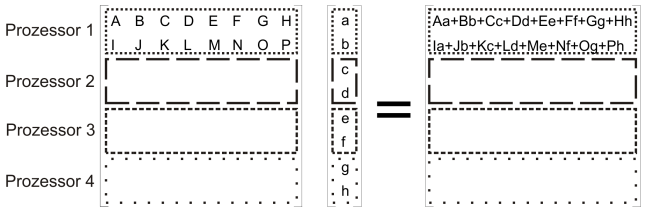

Abbildung 1: Schematische Darstellung der Berechnung des Matrix-Vektor-Produktes verteilt auf mehrere Prozessoren

Matrix-Vektor-Produktes werden sämtliche Zeilen des Vektors auf jedem Prozessor benötigt. Durch entsprechende Kommunikation wird dafür gesorgt, dass allen Prozessoren zur Bildung der Summe die fehlenden Elemente des Vektors zur Verfügung gestellt werden. Nach Ausführung aller Multiplikations- und Additionsoperationen sind die Elemente des neu berechneten Vektors auf die Prozessoren zu verteilen. Die Parallelisierung des Vektor-Vektor-Produktes und der Vektor-Updates ist einfacher zu realisieren. Die Kommunikation zwischen den Prozessoren wird über MPI realisiert. Da die Programmierung auf dieser Kommunikationsebene sehr aufwendig ist, erscheint es zweckmäßig, auf verfügbare Bibliotheken zurückzugreifen.

3.3 Verwendete Softwarebibliothek – PETSc

Es existiert eine Reihe von Softwarepaketen zur parallelen Lösung linearer Gleichungssysteme. Für unsere Zwecke erschien das "Portable Extensible Toolkit for Scientific Computation" (PETSc) naheliegend, da die verwendete Programmierschnittstelle C unterstützt wird und die Mehrzahl der implementierten Gleichungslöser zur Familie der CG-Verfahren gehören. Das Projekt PETSc wird am Argonne National Laboratory entwickelt. Auf der Basis von MPI und numerischen Bibliotheken zur Linearen Algebra werden einfache Funktionen für die Lösung verschiedener partieller Differentialgleichungen in Fortran, C und C++ bereitgestellt. Damit wird die Lesbarkeit des Programmcodes wesentlich verbessert. Ein weiterer Vorteil von PETSc besteht darin, dass sowohl für sequentielle

Programmvarianten (Einzelprozessormaschine) als auch für parallele Programmvarianten im Cluster optimierte Funktionen zur Verfügung stehen.

4 Skalierbarkeitstest

Ein paralleler Algorithmus ist skalierbar, wenn er im Vergleich zur sequentiellen Variante kürzere Rechenzeiten aufweist (Huber, 1997). Im Folgenden soll durch Skalierbarkeitstests die Effizienz paralleler iterativer Gleichungslöser nachgewiesen werden. Zunächst muss eine optimale Kombination aus iterativem Gleichungslöser und Matrix-Vorkonditionierer gefunden werden. Mit dieser Kombination soll anschließend gezeigt werden, ob und in welchem Maße ein paralleler Gleichungslöser auf einem Linux-Cluster skalierbar ist. Diese Tests erfolgen durch Laufzeitmessungen für verschiedene Modellgrößen in Abhängigkeit von der Anzahl der beteiligten Prozessoren.

4.1 Testgrundlagen

Zur Durchführung der Skalierbarkeitstests wurde ein dreidimensionales Leitfähigkeitsmodell verwendet, welches eine vertikale halbunendliche Platte (Dike) in verschiedenen Diskretisierungsvarianten beschreibt (vgl. Abb. 2). Die Anzahl der Blöcke dieses dreidimensionalen Modells wurde zwischen $51 \times 51 \times 24$ und $151 \times 151 \times 74$ variiert, wobei der Gitterabstand immer $\Delta = 10$ m betrug. Das größte Modell führt auf ein lineares Gleichungssystem mit 1 687 274 zu bestimmenden Unbekannten.

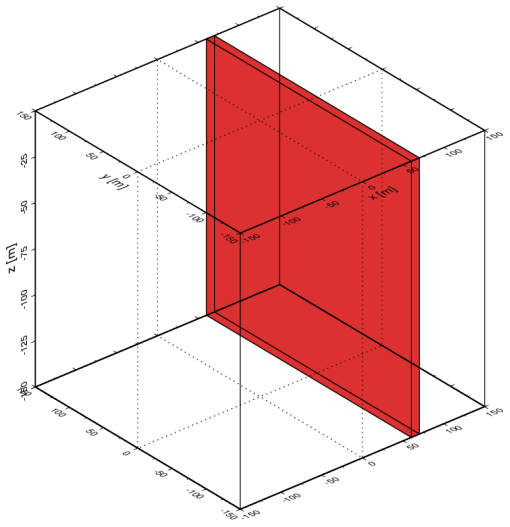

Abbildung 2: Modell der vertikalen halbunendlichen Platte

4.2 Auswahl von iterativem Gleichungslöser und Matrix-Vorkonditionierer

Unter den von PETSc zur Verfügung gestellten Gleichungslösern wurde die Methode der Konjungierten Gradienten als Gleichungslöser ausgewählt, da die schwach besetzte Koeffizientenmatrix des linearen Gleichungssystems symmetrisch und positiv definit ist. In einem vorher durchgeführtem Test hat sich dieses Verfahren erwartungsgemäß durch eine geringe Laufzeit hervorgehoben. Eine effektive Matrix-Vorkonditionierung führt in Verbindung mit einem schnellen iterativen Gleichungslöser zu einer weiteren Verringerung der Gesamtlaufzeit. Um einen geeigneten Matrix-Vorkonditionierer zu

finden, wurden mit einem Modell der Größe $91 \times 91 \times 44$ alle in PETSc verfügbaren Vorkonditionierer hinsichtlich ihrer Gesamtlaufzeit auf einem bis sieben Prozessoren gestestet. Aus den erhaltenen Laufzeiten wurde die Effizienz nach Gleichung (6) berechnet (Abb. 3). Neben sequentiellen stehen in PETSc auch parallele Matrix-Vorkonditionierer zur Verfügung, die eine weitere Effizienzsteigerung ermöglichen. Zu diesen Vorkonditionieren zählen der Block-Jacobi- (BJacobi) und der Additive-Schwarz-Vorkonditionierer (ASM). Beide wenden auf den lokal auf einem Prozessor vorhandenen Teil der parallel verteilten Matrix einen sequentiellen Matrix-Vorkonditionierer (SOR, SSOR oder ILU, zu Details hierzu siehe (Barrett et al., 1994)) an.

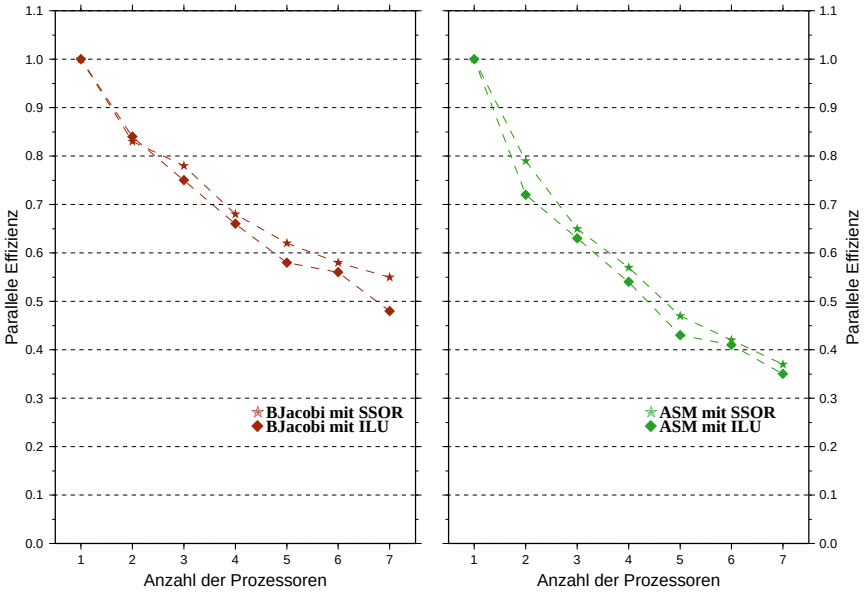

Abbildung 3: Vergleich der verschiedenen Matrix-Vorkonditionierer anhand der Effizienz

Aus Abbildung 3 wird ersichtlich, dass die Block-Jacobi-Vorkonditionierer effizienter als die Additive-Schwarz-Vorkonditionierer sind. Als günstigste Variante eines Matrix-Vorkonditionierers hat sich eine Kombination aus BJacobi als parallelem mit SSOR als sequentiellem Matrix-Vorkonditionierer herausgestellt. Für alle im Folgenden vorgestellten Skalierungtests wurde daher die Methode der Konjugierten Gradienten mit Block-Jacobi/SSOR-Vorkonditionierer benutzt.

4.3 Einfluss der Modellgröße auf die Skalierbarkeit

Um den Einfluss der Modellgröße auf den erreichbaren Laufzeitgewinn beurteilen zu können, wurden Modelle zwischen $51 \times 51 \times 24$ und $151 \times 151 \times 74$ jeweils auf 1 bis 7 Prozessoren gerechnet. In Abbildung 4 ist die benötigte Zeit für einen Iterationsschritt über der Anzahl der Prozessoren aufgetragen. Man beobachtet bei Zunahme der Anzahl der beteiligten PCs für alle Modellgrößen eine Verringerung der benötigten Zeit pro Iterationsschritt.

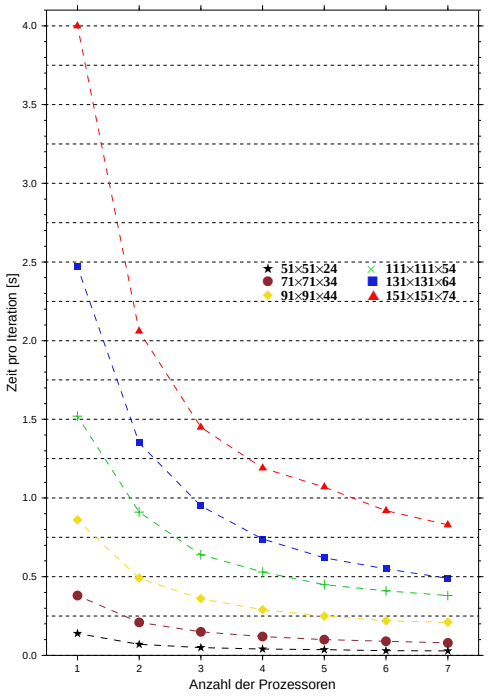

Abbildung 4: Vergleich der Iterationszeit für unterschiedliche Modellgrößen

Ebenso wird deutlich, dass mit zunehmender Modellgröße der erreichte Laufzeitgewinn größer wird. Schließlich kann aus den Laufzeiten der Speedup für die verwendeten Modellgrößen berechnet werden. Es zeigt sich auch hier, dass dieser für größere Modelle höher ist und damit der parallele Algorithmus mit zunehmender Modellgröße besser skalierbar ist.

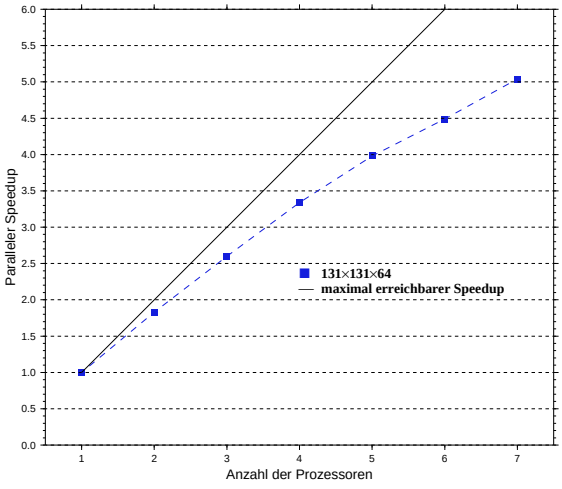

Abbildung 5: Speedup für die Modellgrösse $131 \times 131 \times 64$

Für das Modell der Größe $131 \times 131 \times 64$ wurde der Speedup nach Formel (5) berechnet und in Abbildung 5 über der Anzahl an Prozessoren dargestellt. Beim Einsatz von sieben Prozessoren kann also

die gesuchte Potentialverteilung in einem Fünftel der auf einer Einzelprozessormaschine benötigten Zeit berechnet werden. Der mit diesem Cluster erhaltenen Speedup liegt erwartungsgemäß unter dem theoretisch erreichbaren, da mit zunehmender Anzahl an Prozessoren eine Erhöhung des Kommunikationsaufwandes verbunden ist. Mit größer werdender Prozessorzahl müssen daher Überlegungen angestellt werden, schnellere Netzwerktechnologien einzusetzen.

5 Zusammenfassung

Aus den Ergebnissen der Skalierungstests geht zweifelsfrei hervor, dass für die im Test zur Verfügung stehenden Prozessoren ein signifikanter Speedup zu verzeichnen ist. Beim Einsatz eines heterogenen Linux-Clusters, bestehend aus sieben handelsüblichen PCs, kann die Laufzeit eines parallelen iterativen Gleichungslösers auf 1/5 der auf einer Einzelprozessormaschine benötigten Zeit verringert werden. Es ist damit demonstriert worden, dass mit einem solchen Linux-Cluster effizient parallel gerechnet werden kann. Zieht man die zu erwartende Leistungsfähigkeit kommender Prozessorgenerationen in Betracht, werden damit auch parallelisierte Inversionsalgorithmen realisierbar.

Literatur

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. und van der Vorst, H. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.* Philadelphia, PA: SIAM.

Börner, R.-U., Günther, T. und Käppler, R. (1998). *3D-FD-Modellierung zur Berechnung des Tensors des scheinbaren spezifischen Widerstandes.* In K. Bahr und A. Junge (Hrsg.), *Protokoll 17. Kolloquium "Elektromagnetische Tiefenforschung"* (S. 245-251). Neustadt an der Weinstraße.

Dey, A., und Morrison, H. F. (1979). *Resistivity modeling for arbitrarily shaped three-dimensional structures.* Geophysics, 44(4), 753-780.

Hestenes, M., und Stiefel, E. (1952). *Method of conjugate gradients for solving linear systems.* J. res. Nat. Bur. Standards, 49, 409-436.

Huber, W. (Hrsg.). (1997). *Paralleles Rechnen: Eine Einführung von Walter Huber.* Rosenheimer Straße 145, D-81671 München: R. Oldenbourg Verlag.

Pacheco, P. S. (Hrsg.). (1997). *Parallel Programming with MPI.* 340 Pine Street, Sixth Floor, San Francisco, CA 94104-3205, USA: Morgan Kaufmann Publishers, Inc.

Spitzer, K. (1995). *A 3D finite difference algorithm for DC resistivity modeling using conjugate gradient methods.* Geophys. J. Int., 123, 903-914.