CORBA-basierte verteilte Berechnung medizinischer Bilddaten mit SPM*

Marcel May, Frank Munz, Thomas Ludwig

Lehreinheit Rechnertechnik und Rechnerorganisation/ Parallelrechnerarchitekur
Technische Universität Münchnen, D-80290 München
Email: [mayma|munz|ludwig]@in.tum.de

Zusammenfassung. Der Fortschritt in der Bildgebung und Bildverarbeitung in der Medizin stellt immer höhere Anforderungen an bestehende Rechensysteme. Häufig werden dabei nur einzelne Workstations und PCs benutzt, obwohl die Algorithmen für eine verteilte Berechnung geeignet wären. Im Rahmen des Projekts CIMPTOM[1] (Cluster Based Imaging for Parametric Emission Tomography) wird die Bildvorverarbeitung für die statistische Analyse zeitlicher funktioneller Bildfolgen auf Parallelisierbarkeit untersucht; dabei werden verschiedene Ansätze verglichen.

Schlüsselwörter: Cluster Computing, CORBA, PVM, SPM, Volumen Registrierung

1 Einleitung

Die statistische Bildanalyse setzt voraus, daß alle Bilder stereotaktisch ausgerichtet und normalisiert sind. Die räumliche Ausrichtung versucht, relativ zu einem Referenzbild die linearen Translationen und Rotationen zu berechnen. Diese sechs Transformationsparameter werden durch die Fehlerminimierung mittels der Methode der kleinsten Quadrate ermittelt.

Bei der Normalisierung werden die Bilder an einem Referenzbild normiert, welches den Standard Raum und Form definiert. So gilt für Gehirnaufnahmen meistens die Referenz von Talairach und Tournoux. Neben den linearen Transformationen für die stereotaktische Ausrichtung werden für eine optimale Normierung auch nichtlineare Transformationen ausgeführt.

Beide Schritte der Vorverarbeitung sind für die Parallelisierung geeignet, da alle Bilder einer Folge voneinander unabhängig zu einem Referenzbild berechnet werden. Die sich daraus ergebende grobgranulare Parallelisierungsmöglichkeit mit geringer Kommunikationsfrequenz erlaubt auch die Verteilung auf Ressourcen mit langsamer Anbindung. Als Konsequenz können gewöhnliche, ungenutze Workstations ausgelastet werden, wodurch sich die Gesamtrechenzeit verkürzt.

Die geringe Kommunikationslast erlaubt auch eine freiere Auswahl bei der für die Parallelisierung verwendeten Middleware, so daß ein Vergleich experimenteller Prototypen in bezug auf Design und Eigenschaften interessant ist.

* Gefördert vom Stifterverband für die Deutsche Wissenschaft

Neben der aus dem Bereich des Hochleistungsrechnens kommenden und betriebssystem-nahen Parallel Virtual Machine (PVM) [2] wird als Middleware die abstraktere Common Object Request Broker Architecture (CORBA) [3] eingesetzt, die ein in der Industrie verwendeter Standard ist.

Ausgangspunkt für die Parallelisierung ist die Anwendung Statistical Parametric Mapping Anwendung (SPM) [4]. Die frei verfügbaren Quellen benötigen allerdings die kommerzielle MatLab[5] Laufzeitumgebung. Bei CORBA entschieden wir uns für die Implementierungsvariante Mico [6], einem CORBA 2.3 konformen Object Request Broker der Universität Frankfurt. Die Hardware-Zielplattform ist das Linux PC-Rechencluster der Abteilung für Nuklearmedizin des Klinikums Rechts der Isar in München, welches momentan aus vier 400 Mhz Dual-Pentium II Prozessor Rechnern und einem Ein-Prozessor Rechner besteht. Das parallelisierte SPM wird hier für die Zeitreihenanalyse funktioneller Bildgebung (Positronen-Emissions-Tomographie PET und funktionelles Kernspin fMRI) in der klinischen Forschung eingesetzt.

2 Parallelisierungsaspekte

Die Aufgabe des Realignment ist es, von einem Referenzbild ausgehend die Translationen und Rotationen eines Bildes zu bestimmen. Mit den sich daraus ergebenden sechs Parametern kann eine stereotaktische Korrektur des Bild erfolgen. Bei der Berechnung der Parameter wird nach der Modalität der Daten unterschieden.

Für PET wird dabei zuerst je Lauf die Bewegung zur allerersten Aufnahme berechnet. Da PET Aufnahmen sich durch den Aufnahmezeitpunkt unterscheiden und je nach Rekonstruktionsalgorithmus (z.B. gefilterte Rückprojektion) mit störenden Artefakten versehen sind, wird als nächstes ein Durchschnittsbild von allen im ersten Schritt bereits ausgerichteten Aufnahmen des Durchlaufs berechnet. An diesem rauschärmeren Durchschnittsbild werden nochmals alle Aufnahmen ausgerichtet:

1. Ausrichten der Daten relativ zur ersten Aufnahme je Durchlauf
2. Berechnung des Durchschnittbildes an der neuen Ausrichtung
3. Abschließende Ausrichtung an den Durchschnittbildern je Durchlauf

Handelt es sich um fMRI Bilder, so werden zuerst alle ersten Bilder je Durchlauf zueinander ausgerichtet, gefolgt von der Ausrichtung aller Bilder eines Durchlaufs zum ersten bereits vorher ausgerichteten Bild im selben Lauf:

1. Ausrichten der ersten Aufnahme eines jeden Durchlaufes zueinander
2. Je Durchlauf Ausrichtung der Aufnahmen zur ersten Aufnahme

Die eigentliche Ausrichtung eines Bildes zu einer Referenz geschieht über die Minimierungsmethode der kleinsten Quadrate[7]. Dabei werden iterativ geringe Rotationen und Transformationen auf das Bild angewendet und mittels einer Fehlerfunktion die jeweilige neue Ausrichtung zur Referenz bewertet. Annäherungen werden übernommen, während Abweichungen verworfen werden. Wird ein

Fehlergrenzwert oder eine vorgegebene maximale Iterationstiefe erreicht, gilt das Bild als ausgerichtet und die Iteration terminiert. Im ungünstigsten Fall können keine Ausrichtungsparameter bestimmt werden, weil Bild und Referenz zu geringe Ähnlichkeit aufweisen oder zu sehr stereotaktisch auseinander liegen.

Das Abtasten der Daten erfolgt anfangs mit einer schnellen, aber deshalb auch ungenaueren trilinearen Interpolation. Für die feinere Ausrichtung im Verlauf der Iteration wird zu einer trigonometrischen Interpolation der Bildpunkte gewechselt. Diese ist zwar wesentlich rechenintensiver, liefert aber genauere Werte. Es hat sich durch Profiling verschiedener Implementationen gezeigt, daß das Abtasten der Daten einer der rechenintensivsten Teile des Realignment ist.

Der Ablauf des Realignment erlaubt sowohl eine grobe als auch eine feine Datenparallelisierung. Die grobe Parallelisierung kann die Durchläufe auf mehrere Rechner verteilen. Vorzuziehen ist die feine Parallelisierung; dabei wird die Ausrichtung der einzelnen Bilder zur Referenz auf verschiedene Rechner verlagert. So können auch einzelne Durchläufe effizient berechnet werden.

Die eigentliche Berechnung eines Laufs erfolgt in zwei Schritten. Zuerst erfolgt die Initialisierung, bei der unter anderem vom ersten Bild ausgehend die räumlichen Abtastpunkte festgelegt werden. Diese ca. 6 MByte Initialisierungsdaten benötigen alle Rechnerknoten für die Berechnung der einzelnen Bilder aus diesem Lauf. Die eigentliche Datenmenge hängt von verschieden Faktoren wie z.B. der Dimension des Volumens ab. Der obige Referenzwert bezieht sich auf 128x128x26 Schnittbilder, welche intern mit doppelt genauen Fließkommazahlen (als C/C++ Datentyp double zu 8 Byte) gespeichert werden.

Im zweiten Schritt, der Berechnung der einzelnen Bilder eines Laufs, werden nur noch die Bilddaten übertragen (also 128x128x26x8 = 3.25 MB). Als Ergebnis der Berechnung fällt nur eine 4x4 Transformationsmatrix an; ihre Datengröße von 128 Byte ist vernachlässigbar.

Bei einer durchschnittlichen Berechnungszeit von ca. 6 s je Bild (Intel Celeron 500 MHz) ist diese Kommunikationlast für ein 10 Mb-Netzwerk wenig geeignet, da die Datenübertragung (ca. 6 s) so lange wie die eigentliche Berechnung dauert. Für einen sinnvollen Einsatz ist eine 100 Mb-Vernetzung vorzuziehen, da sich sonst ein ungünstiges Rechen-/Kommunikationsverhältnis ergibt.

Detaillierte Leistungsdaten bezüglich Rechenzeiten und Kommunikationsaufkommen werden im Vortrag vorgestellt.

3 Design und Implementierung

Die PVM Implementierung setzt voraus, daß auf jedem Rechner im Cluster der PVM Dämon läuft. Diese Konfiguration kann PVM aus einer Datei entnehmen. Startet man die PVM Konsole mit dieser Datei, so werden automatisch die Dämone auf den Knoten des Clusters erzeugt.

Die Implementierung besteht aus zwei Programmen, einem Master und einem Worker. Der Master kann für seine Berechnung beliebig viele Worker verwenden. Diese Anzahl kann mittels einem Konfigurationsfile für den Master eingestellt werden. Das PVM Design wird durch folgenden Algorithmus grob beschrieben:

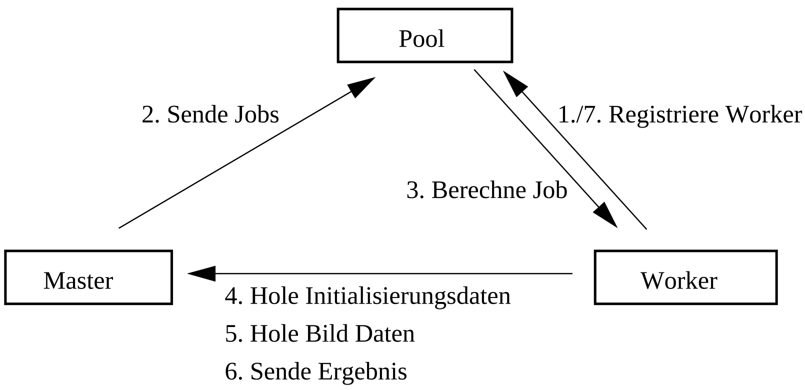

Abb. 1. CORBA Modell

1. Erzeuge Worker-Prozesse auf Rechnerknoten
2. Verschicke per Multicast die Initialisierungsdaten an die Worker
3. Solange Bilder vorhanden :
   (a) Verschicke an jeden Worker ein Bild
   (b) Warte auf Ergebnis von einem Worker
   (c) Verschicke nächstes Bild an diesen Worker
4. Warte auf alle noch rechnenden Worker und deren Ergebnisse
5. Speichere Ergebnisse für SPM und terminiere

Das CORBA Modell besteht aus den drei Objekten Pool, Master und Worker. Diese Objektinstanzen sind als eigene Prozesse implementiert (Abb. 1).

Im Mittelpunkt steht das Pool-Objekt, welches die Rechenaufträge vom Master erhält und auf die Worker im Cluster verteilt. Dieses Objekt registriert sich im CORBA Name Service, so daß Master und Worker sich eine Pool-Objekt Referenz per Name Service bei ihrem Start holen können.

Neben dem Pool-Objekt können mehrere Worker Objekte auf den Knoten existieren, welche sich beim Pool für Rechenaufträge registrieren. Wird von SPM aus ein Realignment angestoßen, so wird aus MatLab heraus ein externes Programm ausgeführt, welches einen Master instanziiert. Dieser Master erzeugt aus dem Realignment-Auftrag mehrere Rechenaufträge, welche an das Pool-Objekt übergeben werden. Wenn alle Resultate der Berechnungen eingetroffen sind, wird sich das Master-Programm beenden und SPM das Ergebnis zurückliefern. Die SPM-Anwendung steht jetzt für weitere Bearbeitungen wieder zur Verfügung.

Ein Rechenauftrag enthält eindeutige Informationen über seine Aufgabe: eine Referenz zum Master selber, damit Ergebnisse zurückgeliefert werden können, und einen eindeutigen Identifikator für die Berechnungsposition (Lauf- und Bildnummer). Erhält das Pool-Objekt Aufträge, schickt es an jeden registrierten Worker einen Auftrag und löscht dessen Registrierung. Das wird solange wiederholt, bis alle Aufträge verteilt sind. Erhält ein Worker einen Auftrag, so werden die Daten direkt vom Master geholt. Gegebenenfalls müssen zuerst noch die Initialisierungsdaten für den Lauf des Auftrags geladen werden. Die Initialisierungsdaten speichert der Worker, um eventuell beim nächsten Rechenauftrag sich das erneute Holen zu sparen. Nach Beendigung der Berechnung schickt das Worker-Objekt das Ergebnis an den Master und registriert sich neu beim Pool-Objekt für weitere Aufträge.

4 Zusammenfassung und Ausblick

Beide Implementierungen habe Vor- und Nachteile. So ist CORBA eine wesentlich komplexere Umgebung, bei der objektorientiert modelliert wird. Daten durchlaufen mehrere Abstraktionsschichten, was Rechenleistung kostet. Andererseits hat man hier auch mehr Gestaltungsmöglichkeiten und das dürfte für größere Projekte interessant sein. Eine nicht verwirklichte Idee beim CORBA-Modell war, die Rechenaufträge mit dem Trading Service zu veröffentlichen statt mittels eines Pool-Objekts zu verteilen. Die Worker könnten sich die Aufträge selber anhand von im Trading Service veröffentlichten Attributen aussuchen. Nebenher spart man sich die Implementation des Pool-Objekts. Die CORBA-Implementation wurde unter dem Gesichtspunkt entworfen, möglichst einfach zu sein. Im Vergleich zu der etwas effizienteren PVM Version hat das verwendete CORBA-Modell den Vorteil, daß die Auftragsverteilung unabhängig von Worker und Master im Pool-Objekt implementiert ist und einfach erweitert oder ausgetauscht werden kann. So könnte man mit geringem Aufwand dynamisch die Verteilungsstrategie für einen optimalen Lastausgleich ändern. Möglich wäre auch eine Implementierung des Pool-Objekts in einer höheren und einfacheren Sprache mit einem CORBA-Mapping wie z.B. Java, da hier wenig auf Performance geachtet werden muss.

Literatur

1. Cimptom Projekt
   http://informatik.tu-muenchen.de/~ludwig/Cimpton/
2. Oak Ridge National Laboratory
   http://www.epm.ornl.gov/pvm/
3. Object Management Group
   http://www.omg.org/
4. The Wellcome Department of Cognitive Neurology
   University College London
   http://www.fil.ion.ucl.ac.uk/spm/
5. The MathWorks Inc.
   http://www.mathworks.com/
6. Mico ORB
   http://www.mico.org/
7. Friston KJ, Ashburner J, Poline JB, Frith CD, Heather JD, Frackowiak RSJ (1995) :
   Spatial Registration and Normalization of Images
   Human Brain Mapping 2:165-189