Parallelisierung eines Programms zur Simulation von Gießvorgängen

Frank M. Thiesing
Fachbereich Mathematik/Informatik
Universität Osnabrück
49069 Osnabrück
E-mail: frank@informatik.uni-osnabrueck.de

ZIAM GmbH
Zentrum für Industrielle Anwendungen Massiver Parallelität
Kaiserstraße 100
52134 Herzogenrath
E-mail: ziam@infoac.rmi.de

Mark Lipinski
MAGMA Gießereitechnologie
Gesellschaft für Gießerei-, Simulations- und Regeltechnik mbH
Werner-Heisenberg-Straße 14
52477 Alsdorf

Abstract. Vorgestellt wird eine Machbarkeitsstudie zur Parallelisierung eines kommerziellen Algorithmus zur Simulation von Aluminium-Gießvorgängen. Ausgangspunkt des Algorithmus sind die Navier-Stokes-Differentialgleichungen für turbulente Flüssigkeitsströmungen, die numerisch durch SOLA-VOF-Algorithmen gelöst werden. Dabei werden sowohl das Finite-Differenzen-Verfahren als auch die Control-Volume-Methode angewendet. Der zentrale Punkt für den effizienten Einsatz eines massiv parallelen Systems ist die optimale Aufteilung der Probleminstanzen in den verschiedenen Phasen des Algorithmus. Im Mittelpunkt steht die Parallelisierung des Jacobi-Verfahrens, mit dem der explizite Solver die Gleichungssysteme löst. Für diesen werden Ergebnisse von Laufzeitmessungen für eine unter PARIX mit FORTRAN 77 implementierte Parallelisierung auf den Parsytec Transputersystemen x'plorer und GC präsentiert.

---

published in: J. Hektor, R. Grebe: *Parallele Datenverarbeitung mit dem Transputer*, pp. 123-135, Springer-Verlag 1994

1. Einleitung

Der Einsatz von Computern bei der numerischen Lösung von physikalischen Problemen hat auf vielen Gebieten zu wesentlichen Fortschritten geführt. Dies gilt insbesondere für die Strömungsmechanik, in der komplizierte mathematische Zusammenhänge nur in sehr einfachen Fällen analytische Lösungen zulassen. Es besteht daher ein großer Bedarf an immer höherer Rechenleistung zur Lösung immer komplexerer Probleme. Konventionelle Hochleistungsrechner erreichen dabei zunehmend wirtschaftliche und physikalische Grenzen. Es ist daher abzusehen, daß zukünftige Supercomputer Parallelrechner sein werden, deren Architekturen zunehmend auf massiver Parallelität beruhen. Diese Entwicklung wird zusätzlich durch die Tatsache unterstützt, daß viele der zu lösenden Probleme eine natürliche Parallelität aufweisen, so daß sie für den Einsatz auf solchen Rechnern prinzipiell geeignet erscheinen. Dies gilt insbesondere für viele Lösungsansätze auf dem hier behandelten Gebiet der Strömungsmechanik.

Bei diesem Teilgebiet der Physik geht es um die Berechnung von Gas- oder Flüssigkeitsströmungen, wie sie in vielen Bereichen der Technik vorkommen. Die Grundlage der Strömungsmechanik sind die Navier-Stokes-Gleichungen. Es handelt sich dabei um ein System von Differentialgleichungen, die für eine Simulation numerisch gelöst werden müssen. Ein Verfahren dazu ist die sogenannte Finite Differenzen Methode (FDM). Dabei wird das Berechnungsgebiet in kleine kubische Teilgebiete, die Control Volumes (CV), unterteilt, auf denen dann die Lösung durch einfache Näherungsverfahren approximiert wird. Wird ein geeignetes Netz von Kontrollvolumen gewählt, so kommt die Approximation der tatsächlichen Lösung sehr nahe.

2. Motivation

Das Programm MAGMASOFT der Firma MAGMA Gießereitechnologie GmbH in Alsdorf ermöglicht die Simulation von Gießvorgängen. Statt aufwendige Experimente in der Gießerei durchführen zu müssen, bietet die Software die Möglichkeit, schon vor dem eigentlichen Gießvorgang die Geometrie des Gußstücks und der Form zu optimieren sowie Randbedingungen wie Füllgeschwindigkeit und Gießtemperatur zu ermitteln. Mit dem Programmpaket werden so aufwendige Gußstücke wie z.B. Aluminium-Zylinderköpfe und Kurbelgehäuse für Fahrzeugmotoren sowie Aluminium-Felgen konstruiert und untersucht.

Als Eingabe erhält die Simulation aus einem programmeigenen CAD-Tool die Geometrie von Gußteil und Form, die anschließend automatisch in kleine Quader diskretisiert werden. Die Anzahl dieser Elemente beträgt bis zu 250 000 pro Gußstück und mehrere Millionen in der Form. Diese Zahl bestimmt im wesentlichen die enorme Rechenzeit der Simulation: Auf den derzeit industrieüblichen 10 MFLOPS-Rechnern kann die Simulation eines einzigen Gießvorgangs eines komplexen Gußstücks mehrere Tage benötigen. Üblicherweise müssen zur Ermittlung der optimalen Gießparameter mehrere Simulationen durchgeführt werden. Wünschenswert für den Benutzer wäre die Berechnung einer Simulation über Nacht oder sogar ‘interaktiv’. Darüber hinaus ist auch eine Verfeinerung des Gitternetzes von Interesse

im Hinblick auf eine detailliertere Beschreibung komplexer Gußteile. Deshalb ist der Wunsch nach einer Steigerung der Rechenleistung um den Faktor 100 durchaus realistisch. Skalierbare Rechenleistung in dieser Größenordnung zu einem vertretbaren Preis bieten nur massiv parallele Systeme.

Das Zentrum für Industrielle Anwendungen Massiver Parallelität ZIAM in Herzogenrath verwendet als Entwicklungssystem einen Parsytec x'plorer mit zur Zeit acht T805 Transputern von INMOS und als Test- und Produktionsmaschine den Parsytec GCel-1024 an der Universität Paderborn mit 1024 T805. Als Entwicklungsumgebung dient in diesem Projekt PARIX 1.2 mit FORTRAN 77, um möglichst große Teile des sequentiellen Codes unverändert übernehmen zu können.

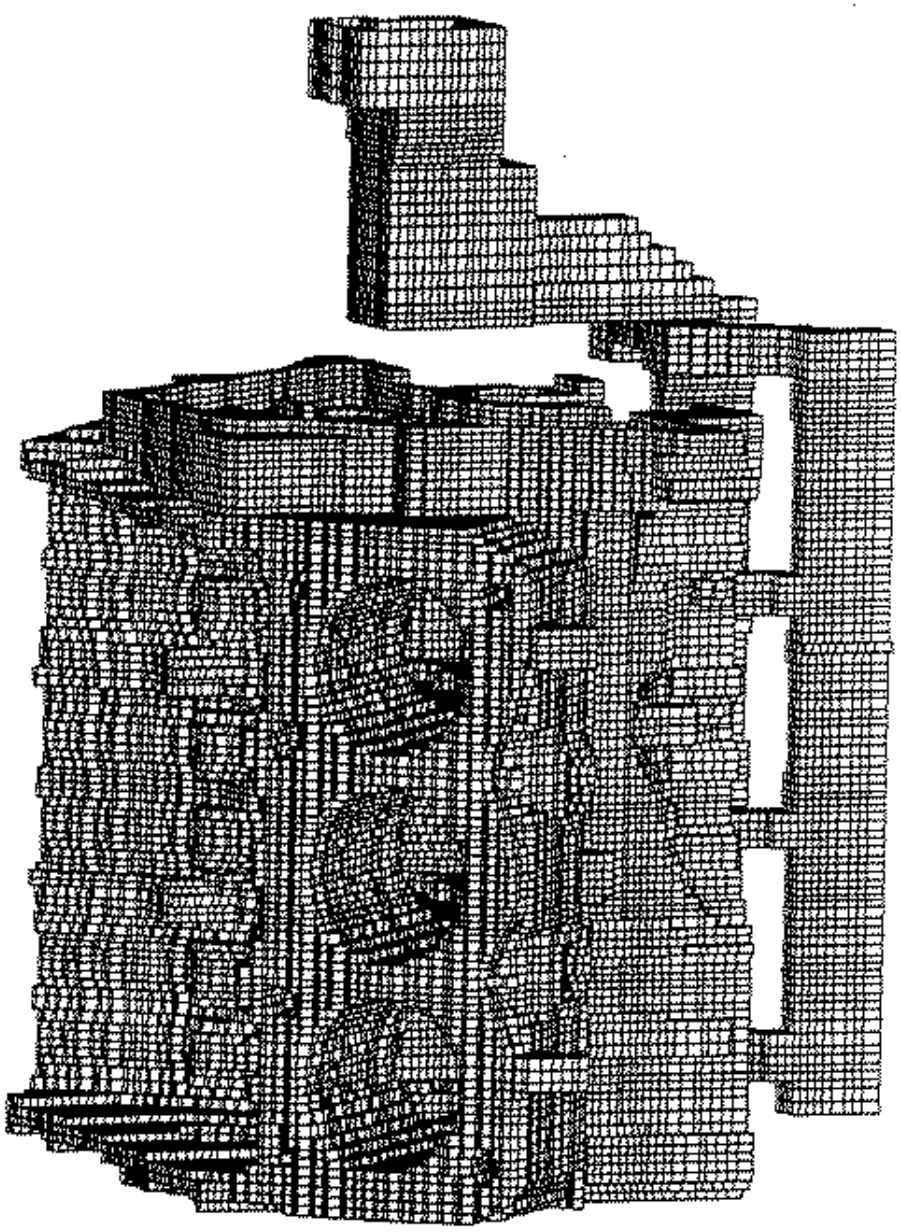

Abbildung 2.1: Vernetzung eines V6-Kurbelgehäuses mit ca. 2,4 Millionen Elementen

3. MAGMASOFT

Das Softwarepaket MAGMASOFT gliedert sich im wesentlichen in vier Teile:

3.1. MAGMAPRE

Zum Preprocessing gehört die Eingabe des zu gießenden Teils und der umgebenden Formbox mit Hilfe eines CAD-Programms. Neben der reinen Geometrie werden Angaben über Zuflüsse (Inlet) und Entlüftungen sowie über die verschiedenen Materialien, für die Form z.B. Sand oder Stahl, für das Gießteil die Art der Legierung, gemacht. Außerdem können Temperaturen und Füllgeschwindigkeiten eingestellt werden. Für die anschließenden Berechnungen wird die gesamte Formbox von einem Netzgenerator automatisch in kleine Quader zerlegt. Dies geschieht, indem in allen drei Dimensionen achsenparallele Ebenen durch die Formbox gelegt werden (s. Abbildung 2.1). Die Feinheit des entstehenden Gitters kann dadurch beeinflußt werden, daß dem einzelnen Volumenelement Beschränkungen bezüglich seiner Ausdehnung auferlegt werden. Das Ergebnis ist eine Menge von orthogonalen Volumenelementen (Quadern), die aber nicht alle gleich groß sind, da der Netzgenerator z.B. bei Rundungen des Gießteils automatisch soweit wie zugelassen verfeinert. Jedes dieser Control Volumes wird eindeutig zugeordnet: Entweder zur Form, zum Inlet oder zum Gießteil. Durch diese Diskretisierung ergeben sich bis zu 2,5 Millionen Volumenelemente, von denen etwa 10% zu füllen sind.

3.2. Füllsimulation

Die Eingabedaten aus dem CAD-Programm dienen der Füllsimulation als Input. Aufgabe dieses Programmteils ist es, den Füllvorgang iterativ numerisch zu simulieren. Dazu werden in einer großen Anzahl von Zeitschritten (1000-10 000) jeweils physikalische Werte in allen Volumenelementen berechnet. So z.B. Druck, Temperatur, Fließgeschwindigkeit und -richtung. Zu vorgegebenen Breakpoints ist die Ausgabe des Status quo in eine Datei zum Zwecke des Postprocessings möglich. Ziel der Füllsimulation ist neben der Ausgabe, ob das Teil unter den vorgegebenen Bedingungen überhaupt zu gießen ist oder ob die freie Oberfläche der Schmelze vorher erstarrt, die Lieferung der physikalischen Werte in allen Control Volumes am Ende des Füllvorgangs. Die Programmlaufzeit zur Simulation eines Füllvorgangs beträgt auf einer Maschine mit ca. 10 MFLOPS bis zu 5 Tagen. Dieser Vorgang muß mit veränderten Parametern wiederholt werden, bis das Ergebnis der Simulation den Erwartungen entspricht.

3.3. Erstarrungssimulation

Die Ergebnisse aus der Füllsimulation, insbesondere das Temperaturfeld am Ende des Füllvorgangs, dienen der anschließenden Erstarrungssimulation als Eingaben. Die Laufzeit für eine Simulation beträgt etwa einen Tag.

3.4. MAGMAPOST

Zum Postprocessing gehört die Darstellung der unterschiedlichen physikalischen Bedingungen im Gießkörper und der umgebenden Form. Dazu werden die zu vorgegebenen Zeiten geschriebenen Dateien herangezogen. Die Darstellung erfolgt in dreidimensionaler Projektion in der CAD-Umgebung mit der Möglichkeit, Details zu betrachten und Schnittebenen zu definieren.

4. Füllsimulation

Die Füllsimulation steht wegen ihres enormen Rechenzeitbedarfs im Mittelpunkt der Betrachtungen. Die physikalischen Probleme zu Flüssigkeitsdynamik und Wärmetransport sind mathematisch durch Differentialgleichungen formuliert [Lip92].

Der numerische Algorithmus arbeitet nach der Methode der Control Volumes (CV). Verwendet wird die Finite Differenzen Methode (FDM). Der Lösungsalgorithmus wird als SOLA-VOF (Solution of linear algorithms - Volume of fluid) bezeichnet [Hit79].

In jedem CV der Formbox existiert ein Wert für die Temperatur. In den gefüllten CVs berechnet der Algorithmus zusätzlich Druck, Geschwindigkeit und Füllmenge in Prozent, außerdem in Abhängigkeit von der Temperatur Dichte, Viskosität und Wärmeleitfähigkeit. Eine gesonderte Behandlung erfahren die CVs an der sogenannten freien Oberfläche der Schmelze (s. Abbildung 4.1). Aus Gründen der numerischen Stabilität wird in so kleinen Zeitschritten gerechnet, daß die freie Oberfläche höchstens die nächste Schicht von CVs im zu füllenden Volumen erreicht.

Im folgenden wird die Arbeitsweise des Füllalgorithmus detaillierter dargestellt.

4.1. Füllalgorithmus

Die anschließenden Schritte finden für jeden der 1000-10 000 Zeitschritte statt:

4.1.1. Berechnung der Länge des Zeitschritts

Da in jedem Zeitschritt aus numerischen Gründen die freie Oberfläche der Schmelze nur höchstens eine CV-Schicht fortschreiten darf, muß anhand der Parameter in den bereits gefüllten Metallzellen die Länge des nächsten Zeitschritts berechnet werden.

4.1.2. Predictor Schritt

Aus den Geschwindigkeiten des vorherigen Zeitschritts wird die Beschleunigung in jeder Metallzelle berechnet und durch die Länge des Zeitschritts eine neue vorläufige Geschwindigkeit "vorhergesagt". Bei dieser FDM-Berechnung werden die Geschwindigkeiten in den benachbarten CVs verwendet. Dieser Schritt 2 nimmt etwa 3-4% der CPU-Zeit in Anspruch.

4.1.3. Corrector Step

Die in Schritt 2 berechneten Geschwindigkeiten erfüllen nicht die Kontinuitätsgleichung in den CVs. Deshalb wird hier ein Druckkorrekturterm berechnet, mit dem das neue Druck- und Geschwindigkeitsfeld richtig angegeben werden kann. Zur Berechnung in einem CV werden die Werte aus den sechs benachbarten Zellen vorne (front), hinten (back), links (west), rechts (east), oben (north) und unten (south) benötigt. Es ergibt sich ein Gleichungssystem mit einer Gleichung pro gefüllter Metallzelle in Abhängigkeit von den Werten in den sechs Nachbarzellen. Dieses gilt es möglichst effizient zu lösen, wobei die numerische Stabilität und die schlechte Kondition der Matrix zu beachten sind.

Der vorliegende sequentielle Algorithmus verwendet ein dreistufiges Verfahren:

Zunächst wird mit Hilfe des Jacobi-Iterationsverfahrens versucht, die Lösung des gegebenen LGS zu bestimmen. Dies gelingt in etwa 60% der Fälle. Dieser Schritt wird als expliziter Löser bezeichnet und in Abschnitt 5.1 näher beschrieben.

Wenn die Konvergenz zu schlecht ist, kommen ein ebenfalls auf der Jacobi-Iteration beruhendes implizites Verfahren und zusätzlich ein sogenanntes "mass rebalancing" zum Einsatz.

Insgesamt benötigt dieser 3. Schritt 80-85% der CPU-Zeit des gesamten Füllalgorithmus.

4.1.4. Temperaturberechnung

Dieser Schritt wird nur in jedem 30.-50. Zeitschritt durchgeführt. Aus der Energiegleichung wird die Temperatur in den CVs berechnet. Dabei wird zum einen in allen (!) CVs der Formbox die Diffusion der Temperatur berechnet. Dabei spielt auch die Wärmeabstrahlung der freien Oberfläche an die Wände eine Rolle. Dies ist die einzige Stelle im Algorithmus, wo auch auf die CVs in der umgebenden Box zugegriffen wird.

Zusätzlich wird nur für die gefüllten Metallzellen die Konvektion der Temperatur mit einem expliziten Lösungsverfahren berechnet. Dieser Schritt beansprucht etwa 3% der CPU-Zeit.

4.1.5. Berechnung der freien Oberfläche

Es werden Randbedingungen für die CVs der freien Oberfläche betrachtet. Dazu ist die Information der benachbarten CVs notwendig. Anschließend wird die Entwicklung der freien Oberfläche berechnet. Dazu gehört auch, wie weit ein CV mit Metall gefüllt ist.

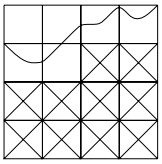

Abbildung 4.1: (Teilweise) gefüllte und leere Zellen an der freien Oberfläche

An dieser Stelle ist im Programm ein umfangreiches Updaten der Listen erforderlich, da CVs der freien Oberfläche durch das Nachfließen des Metalls zu gefüllten Zellen geworden sind und zuvor leere CVs in die Liste der freien Oberfläche aufgenommen werden müssen. Es ist aber auch möglich, daß durch Wellenbewegungen der Schmelze CVs der freien Oberfläche wieder zu leeren Zellen werden und bereits gefüllte Zellen wieder geleert werden.

Obwohl sich diese Berechnungen im Bereich der wenigen tausend CVs der freien Oberfläche abspielen, werden doch 10-15% der CPU-Zeit hier verbraucht.

4.1.6. Output

In jedem Zeitschritt erfolgt die Ausgabe, zu wieviel Prozent die Form gefüllt ist. Darüberhinaus werden zu den vorgegebenen Breakpoints das Temperaturfeld in der gesamten Formbox sowie Druck und Geschwindigkeit in den Metallzellen ausgegeben.

4.1.7. STOP/NEXT

Ist die Form ganz gefüllt, wird der Füllalgorithmus beendet. Eine vorzeitige Abbruchbedingung ist das Erstarren der freien Oberfläche, wodurch ein weiteres Füllen unmöglich ist. Sonst geht es mit der Berechnung des nächsten Zeitschritts weiter.

Der gesamte Füllalgorithmus erlaubt eine Aussage darüber, ob das Gießen unter den vorgegebenen Bedingungen möglich ist, und wie die Temperaturverteilung, mit der die Erstarrungssimulation beginnt, am Ende des Füllens aussieht.

5. Parallelisierungsmöglichkeiten

Die Arbeiten zur Parallelisierung beginnen beim Jacobi-Verfahren, um den rechenintensiven expliziten Löser möglichst effizient zu parallelisieren. Dieser arbeitet nur auf den Elementen, die im aktuellen Zeitschritt vollständig mit Metall gefüllt sind (und auf den Zellen am Rand des Gußstücks), um hier Druck und Geschwindigkeit zu bestimmen. Aufgrund des zeitlichen Verlaufs des Füllens sind dies zunächst nur wenige Zellen, wobei in jedem Zeitschritt maximal eine Schicht von Elementen (im Mittel ca. 20-30) dazukommt.

Das Problem der Domain Decomposition des geometrisch komplexen Gußstücks ist also der Schlüssel zu einem effizienten parallelen Algorithmus. Werden wegen einer zu statischen Verteilung der Elemente zunächst nur wenige Prozessoren beschäftigt, so geht dieses zu Lasten des Speedups. Außerdem erfordert eine statische Aufteilung mit guter Lastverteilung Kenntnisse über den Strom des Metalls im Gußstück, der erst berechnet werden soll. Eine solche Aufteilung der Elemente unter den Prozessoren kann deshalb nicht die optimale Lösung sein.

Eine dynamische Verteilung der Elemente unter den Prozessoren in der Reihenfolge ihrer Füllung erscheint unter dem Zwang einer gleichmäßigen Lastverteilung angebrachter. Dabei muß allerdings ebenfalls die Geometrie des Gußstücks berücksichtigt werden, um zusammenhängende Gebiete von Elementen möglichst auf einem Prozessor zu berechnen. Dieses ist notwendig, um den Kommunikationsaufwand zum Austausch der

Randbedingungen der verteilten Speicherbereiche in Grenzen zu halten.

Bei der Analyse des sequentiellen Programms ergibt sich für ein Gußstück mit ca. 200 000 Elementen ein Speicherbedarf von etwa 20 MByte. Die gesamte Form hat ca. zehnmal soviele Elemente und benötigt zusätzlich 60-70 MByte. Der Versuch, lediglich den expliziten Löser zu parallelisieren scheitert daran, daß in jedem Zeitschritt etwa 12 MByte zwischen Frontend- und Parallelrechner übertragen werden müßten. Außerdem wären damit nur ca. 60% des Algorithmus parallelisiert, was nach Amdahls Gesetz den theoretisch möglichen Speedup begrenzt. Amdahls Gesetz besagt:

$$\text{Speedup} = \frac{1}{f + \frac{1-f}{p}} < \frac{1}{f}\,.$$

Dabei ist p die Anzahl der eingesetzten Prozessoren und f der sequentielle Anteil eines parallelen Algorithmus.

Eine Parallelisierung kann deshalb nur effizient sein, wenn fast der gesamte Teil des Füllalgorithmus auf dem massiv parallelen System läuft und die Kommunikation mit dem Frontend begrenzt ist.

Der hohe Rechenbedarf für das Gußstück und der hohe Speicherbedarf für die Elemente in der umgebenden Form legen es nahe, die vollständige Parallelisierung der gesamten Füllsimulation zwischen Frontend (Workstation) und Parallelrechner aufzuteilen: Während der Parallelrechner die physikalischen Werte in den Elementen des Gußstücks berechnet, werden gleichzeitig die Temperaturwerte in den umliegenden Elementen von dem Frontendrechner aktualisiert. Dazu ist etwa alle fünfzig Zeitschritte ein Austausch von ca. 3 MByte nötig. Der Vorteil dieses Vorgehens besteht darin, daß die kleinen Transputerknoten (z. Zt. 4 MByte) nicht durch die 60-70 MByte für die Form belastet werden, die für eine große Workstation kein Problem darstellen. Zusätzlich wird so auch algorithmisch die Parallelität zwischen Parallelrechner und Frontend ausgenutzt (s. Abbildung 5.1).

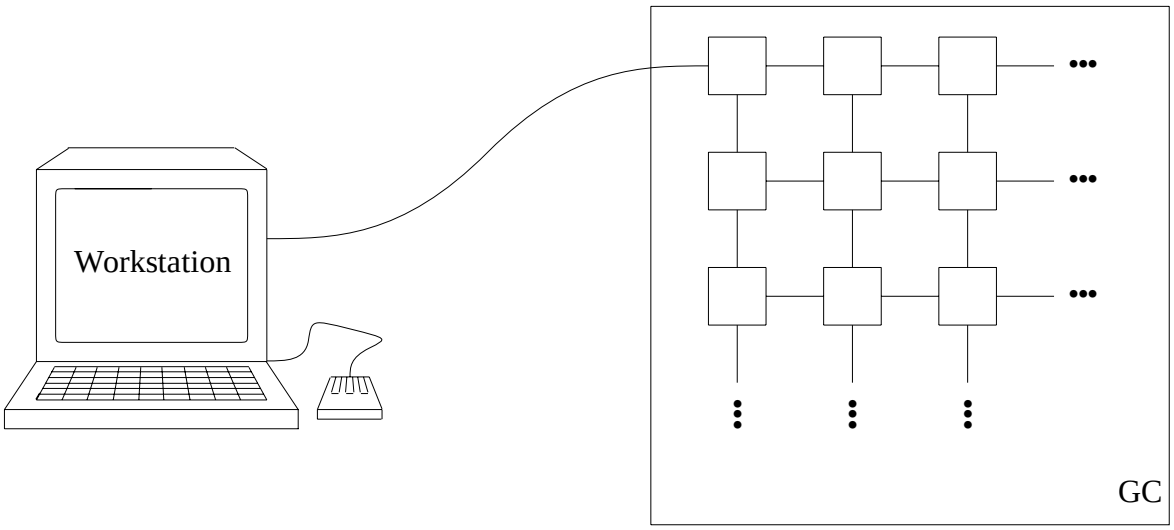

Abbildung 5.1: Frontend Workstation und Transputernetzwerk GC

5.1. Das Jacobi-Verfahren

Mit Hilfe des Jacobi-Verfahrens werden in jedem Zeitschritt die neuen Werte von Druck und Geschwindigkeit in den bereits gefüllten Metallzellen berechnet. Der Predictor-Schritt (s. Abschnitt 4.1.2) berechnet aus den Bewegungsgleichungen ein neues Geschwindigkeitsfeld. Dieses erfüllt nicht notwendigerweise die Kontinuitätsgleichung; die Massedifferenz (*rmass*) in mindestens einer Zelle ist ungleich Null. Von der Massedifferenz wird eine Druckkorrektur *dp* berechnet. Mit dieser werden das Druck- und Geschwindigkeitsfeld im Corrector-Schritt korrigiert (s. Abschnitt 4.1.3). Für die Berechnung der Druckkorrektur wird das Jacobi-Verfahren angewendet.

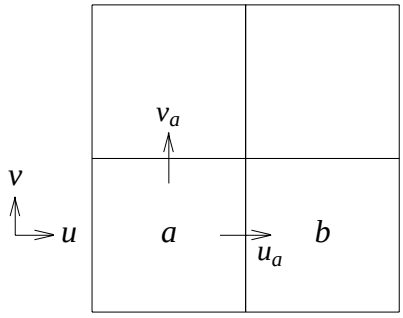

Abbildung 5.2: Physikalische Werte eines Control Volumes

Gegeben in jedem Iterationsschritt:

p_a	Druck im CV *a*
$u_{a\ \mathrm{alt}}$, $v_{a\ \mathrm{alt}}$, $w_{a\ \mathrm{alt}}$	alte Geschwindigkeiten in drei Dimensionen bei CV *a*
due_a, dvn_a, dwb_a	Druckkorrekturkoeffizienten im CV *a*

Berechnet werden:

$rmass_a$	Massedifferenz in CV *a* (aus den alten Geschwindigkeiten)
dp_a	Druckkorrektur in CV *a* (aus der Massedifferenz)
$u_{a\ \mathrm{neu}}$, $v_{a\ \mathrm{neu}}$, $w_{a\ \mathrm{neu}}$	neue Geschwindigkeiten in drei Dimensionen bei CV *a*

Es gilt nämlich:

$$u_{a\ \mathrm{neu}} = f\,(u_{a\ \mathrm{alt}}, dp_a, dp_b)$$

Der iterative Algorithmus terminiert, wenn in allen gefüllten Zellen die Massedifferenz unter eine gewisse Genauigkeit gefallen ist. Für die Berechnung von Druck und Geschwindigkeit in jeder Zelle werden lediglich die Werte der sechs Nachbarzellen benötigt.

Die Information über die Nachbarschaft wird in den Vektoren *iw*, *ie*, *is*, *in*, *if*, *ib* gehalten, die den Index des jeweiligen Nachbarn beinhalten.

5.2. Parallelisierung des Jacobi-Verfahrens

Zur Parallelisierung wird die Domain des Gußstücks unter den Prozessoren verteilt. Da zur Berechnung der neuen Werte in einer Zelle die Daten der direkten Nachbarn benötigt werden, ist nach jedem Iterationsschritt ein Randwertaustausch notwendig, da sich die Druckdifferenzen geändert haben (s. Abbildung 5.3).

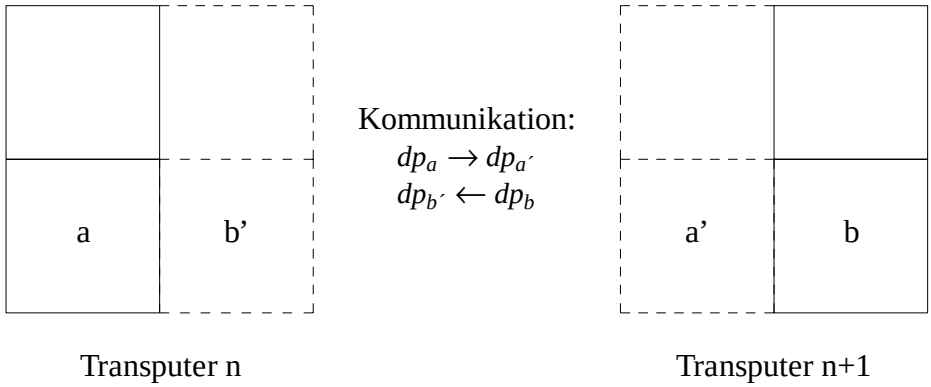

Abbildung 5.3: Randwertausgleich

Hinderlich bei der Parallelisierung ist die für die sequentielle Implementierung in FORTRAN eingeführte indirekte Indizierung der Felder.

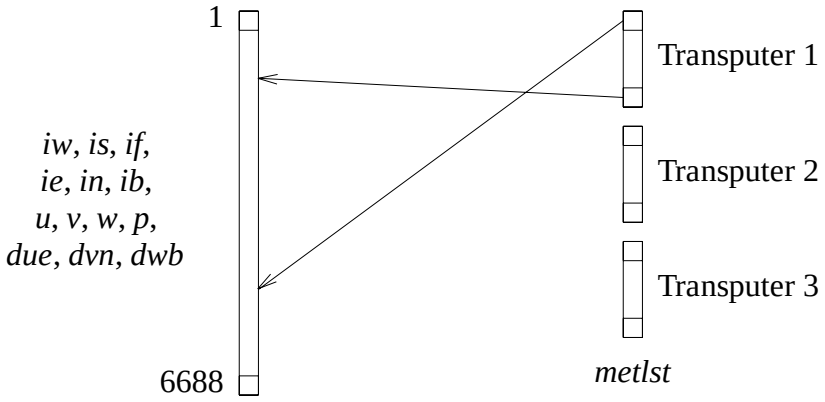

Abbildung 5.4: Indirekte Adressierung

Die Zahl gefüllter Metallzellen steigt in jedem Zeitschritt. Für diese wird ein Feld *metlst* gehalten, das lediglich die Indizes für die globalen Listen enthält, in denen die Informationen für alle zufüllenden CVs des Gußstücks stehen (s. Abbildung 5.4).

6. Beispiel für Messungen

Als Testbeispiel wird ein Gußstück in einer Form mit $22 \times 22 \times 34 = 16\,456$ CVs gerechnet. Für die Jacobi-Iteration interessieren davon nur das Gußstück mit einer Schicht von der Form als Rand, in diesem Fall $16 \times 16 \times 26 + 32 = 6688$ CVs (s. Abbildung 6.1).

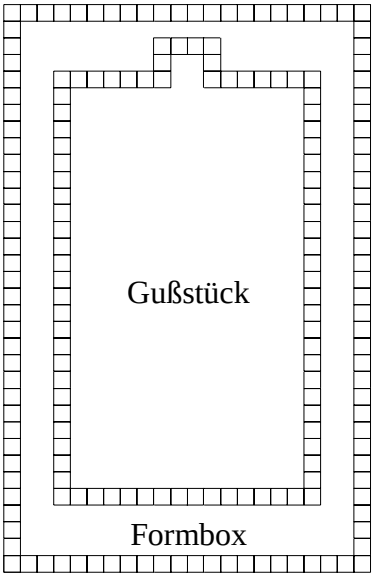

Abbildung 6.1: Gußstück für Performance-Messungen

Von diesen sind 4522 bereits ganz gefüllt, nur in den oberen beiden Schichten unter dem Inlet sind noch leere Zellen sowie die der freien Oberfläche. Die Verteilung der Zellen auf die Transputer erfolgt in dieser ersten Parallelisierung schichtweise (s. Abbildung 6.2), um nur in einer Dimension den Randwertausgleich durchführen zu müssen.

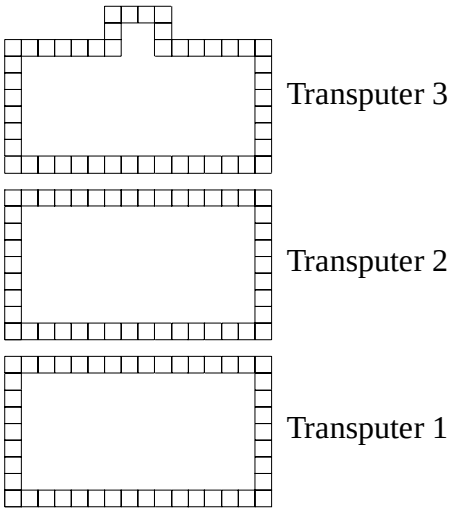

Abbildung 6.2: Aufteilung des Gußstücks unter 3 Transputern

Wegen der geringen Anzahl von Schichten in diesem Beispiel und weil die oberen Schichten nicht ganz gefüllt sind und deshalb dort weniger zu berechnen ist, werden die Messungen auf maximal 24 Transputern und Teilern davon durchgeführt.

7. Ergebnisse

Die Parallelisierung des sequentiellen FORTRAN- Programms wird unter dem Betriebssystem PARIX 1.2 durchgeführt. Als Entwicklungsumgebung steht ein Parsytec x'plorer mit acht T805 Transputern zur Verfügung. Für die Laufzeitmessungen wird der GCel-1024 mit 1024 T805 an der Universität Paderborn eingesetzt. Als sequentielle Referenz-Maschine wird eine SUN Sparc 10 herangezogen. Folgende Zeiten ergeben sich bei der Berechnung zur Lösung des Gleichungssystems mit dem Jacobi-Verfahren in einem Zeitschritt (s. Tabelle 7.1 und Abbildung 7.1).

# Transputer	1	2	3	4	6	8	12	24
Zeit (sek)	440.0	236.8	163.5	125.0	87.0	68.2	50.2	34.8
Sparc 10 (sek)	68.7							

Tabelle 7.1: Messwerte für das Beispiel

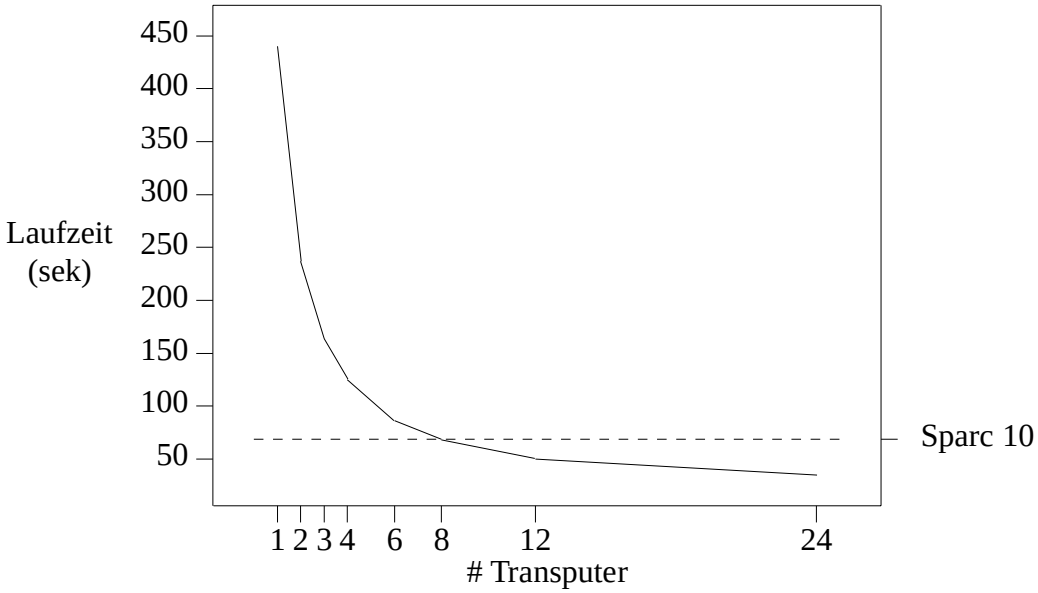

Abbildung 7.1: Vergleich zwischen Sparc 10 und bis zu 24 Transputern

8. Zusammenfassung

Das Vorprojekt zur Parallelisierung der Simulationssoftware MAGMASOFT soll in einer Machbarkeitsstudie klären, ob der Algorithmus der Füllsimulation effizient und nutzbringend parallelisiert werden kann.

Die Arbeiten dazu beginnen beim rechenintensivsten Programmstück, dem Jacobi-Verfahren zur Lösung großer Gleichungssysteme der Strömungssimulation. Der Parallelisierung auf dem Transputersystem kommt insbesondere die Laufzeitumgebung PARIX sehr entgegen, die die Übernahme großer Teile des sequentiellen FORTRAN-Codes erlaubt und dadurch die Implementation erheblich beschleunigt. Dennoch sind in wesentlichen Teilen des Programms algorithmische Umstellungen und Neucodierungen angeraten, um die Effizienz des Programms im Hinblick auf das Ziel einer massiven Beschleunigung der Ausführung weiter zu steigern.

Insbesondere die indirekte Adressierung der Felder im Ausgangsprogramm erweist sich bei der Parallelisierung wegen der notwendigen Domain Decomposition als hinderlich. Hier sollte das Programm modifiziert werden, was einen nicht unerheblichen Aufwand darstellt.

Die Laufzeitmessungen der ersten parallelen Version des expliziten Lösers zeigen die Möglichkeit, durch den Einsatz der parallelen Datenverarbeitung industrielle Anwendungen erheblich beschleunigen und durch weitere Optimierungen des parallelen Algorithmus effizient auf das massiv parallele System portieren zu können.

9. Literaturverzeichnis

[Lip92] D.M. LIPINSKI, W. SCHAEFER, E. FLENDER: *Numerical Modelling of the Filling Sequence and Solification of Castings*; MAGMA Gießereitechnologie GmbH, Alsdorf 1992

[Hir79] C.W. HIRT, B.D. NICHOLS: *Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries*; Los Alamos Scientific Laboratory 1979, Journal of Computational Physics 1981

[Mor] K. MORGAN, J. PERIAUX, F. THOMASSET: *Analysis of Laminar Flow over a Backward Facing Step*; Notes on Numerical Fluid Mechanics Volume 9, Vieweg&Sohn

[Rod89] G. RODRIGUE: *Parallel Processing for Scientific Computing*; Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing, Los Angeles 1987, SIAM 1989

[Eva91] D.J. EVANS: *Parallel Computing 90/91*; Proceedings on the International Conference on Parallel Computing, North-Holland 1991

Talk presented at: 5. Transputer-Anwender-Treffen TAT/WTC ’93, Aachen, 20.-22. September 1993