Institut für Informatik — TU München WS 2009/10
Scientific **C**omputing in **C**omputer **S**cience
Prof. Dr. M. Bader
Dipl.Inf. T. Auckenthaler, Dipl.Inf. W. Eckhardt

Masterpraktikum Scientific Computing (High Performance Computing) Übungsblatt 1: Vektorisierung

Zur Übung am 20.10.2009

Aufgabe 1 „Automatische Vektorisierung“(1 Punkte)

Lesen Sie Kapitel 21 „Using Parallelism: Automatic Vectorisation“der **Intel C++ Compiler User and Reference Guides** und beantworten Sie die folgenden Fragen:

- Welche Arten von Schleifen können automatisch vektorisiert werden?
- Welche Datentypen und welche Operationen dürfen in Schleifen verwendet werden, damit diese vektorisierbar sind?
- Welche Datenabhängigkeitsanalysen führt der Compiler durch?
- Welche Einfluss hat die Art und Weise, wie ein Programm geschrieben ist, auf die Vektorisierbarkeit?
- Durch welche sprachspezifischen Konstrukte können Sie den Compiler bei der automatischen Vektorisierung unterstützen?
- Welche Schleifenoptimierungen kann der Compiler vornehmen, um Schleifen effizient zu vektorisieren bzw. effizient Pipelining zu nutzen?

Aufgabe 2 „Abhängigkeitsanalyse und Vektorisierung“(1 Punkte)

Der Programmcode `vector.c` enthält mehrere Funktionen, die Schleifen enthalten. Manche können als Vektoroperation ausgeführt werden. Geben Sie an, welche Programmteile vektorisierbar sind. Geben Sie eventuelle Bedingungen an, an die die Vektorisierung geknüpft ist. Versuchen Sie, soweit möglich, durch Manipulation der Schleifen bzw. durch Compilerdirektiven die Vektorisierung auf dem Rechner **ia64lx3** zu erreichen. Verwenden Sie den Compiler der Firma Intel in der Version 10. Ziehen Sie auch das Kapitel über Vektorisierung in **Intel C++ Compiler User and Reference Guides** zu Rate.

- Geben Sie für jede Funktion Ihre Einschätzung bezüglich der Vektorisierbarkeit der enthaltenen Scheifen an!
- Welche Ausgabe erhalten Sie vom Compiler? Welche Scheifen werden vektorisiert?
- Können Sie durch einfache Manipulation der Schleifenstruktur, des Schleifenkörpers oder durch Compilerdirektiven die Vektorisierung weiterer Schleifen erreichen?
- Sichern Sie die Ausgabe des Compilers ohne und mit Ihren Änderungen in einer Datei!
- Beachten Sie die Kommentare und Hinweise im Quellcode!

Aufgabe 3 „SSE2 Intrinsic Functions“(2 Punkte)

Entwerfen und implementieren Sie ein Programm, das jedes Element des Vektors $a[i]$ durch den Wert b teilt und einen Vektor $fc[i]$ mit den ganzzahligen Quotienten sowie einen Vektor $mc[i]$ mit den jeweiligen Divisionsresten berechnet, sodass $a[i] = b * fc[i] + mc[i]$ gilt. Dabei sollen alle Werte als Gleitpunktzahlen mit doppelter Genauigkeit abgespeichert werden. Initialisieren Sie den Vektor $a[i]$ mit zufälligen Zahlen.

- Implementieren Sie das Programm in C. Versuchen Sie, das Programm durch den Intel-Compiler automatisch vektorisieren zu lassen. Welche MFLOP-Rate können Sie auf der lx64ia3 erreichen?
- In wie weit können Sie mit anderen Compilern (GNU gcc, PGI) eine automatische Parallelisierung erreichen? Vergleichen Sie die MFLOP-Raten!
- Implementieren Sie Ihr Programm nun mit Hilfe der `SSE2 Intrinsic Functions`. Welche Performance können Sie nun erzielen?
- Messen Sie die FLOP-Rate der beiden Programme auf dem Lehrstuhlrechenr atsccs30! Übersetzen Sie das reine C-Programm dabei einmal mit dem GNU gcc sowie mit dem Intel Compiler! Hinweise:
  - Verwenden Sie die Funktion `_mm_cvttpd_epi32()`.
  - Beachten Sie die entsprechenden Kapitel in **Intel C++ Compiler User and Reference Guides**
  - Auf den Lehrstuhlrechnern ist der Intel Compiler in der Version 9 installiert. Hier müssen Sie die Option `-msse2` spezifizieren, um eine Vektorisierung mittels SSE2 zu erreichen.

Aufgabe 4 „Vektorisierung über die Anzahl der Probleme“(2 Punkte)

Mit dem Programm `gauss.c` steht Ihnen eine Implementierung der Gauß-Elimination ohne Pivotsuche mit Rücksubstitution zur Verfügung. Mit diesem Algorithmus können Sie ein lineares Gleichungssystem der Ordnung n explizit lösen. Hat das Gleichungssystem die Form $A\vec{x} = \vec{b}$, so ist A die Koeffizienten-Matrix, $\vec{b}$ ist die rechte Seite und $\vec{x}$ ist der Lösungsvektor.
Skizzieren Sie kurz die Funktionsweise des Algorithmus in der gegebenen Form!
Es sollen 2000 Gleichungssysteme mit der Ordnung $n = 3$ gelöst werden. Das Lösen von kleinen Gleichungssystemen ist auf einem Vektorrechner bzw. auf einem PC oder einer Workstation mit dem gegebenen Algorithmus nicht effizient! Warum?

Es ist Ihre Aufgabe, den Algorithmus so zu verändern, dass ein optimal vektorisiertes Programm vorliegt (Stichwort: „Vektorisierung über die Anzahl der Probleme“). Überlegen Sie sich eine geeignete Datenstruktur! Messen Sie die MFLPOS-Rate! Welche Leistungssteigerung können Sie auf dem Rechner **lx64ia???** erzielen?

Anmerkung zu den Compilern:

- Der PGI Compiler wird durch den Befehl `module load ccomp/pgi/7.1` verfügbar.
- Empfohlene Flags: `-O2 -tp p7 -fastsse -Minfo=loop -Mneginfo=loop`
- Das „User Manual“ zum `pgcc` kann bei der Portland Group heruntergeladen werden.
- Das User Manual des Intel Compilers finden Sie auf der Internet-Seite der Firma Intel.
- Der gcc-4.3.3 wird durch den Befehl `module load gcc` verfügbar.
- Informationen zur Benutzung des LRZ-Linux-Cluster erhalten Sie unter *http://www.lrz-muenchen.de/services/compute/linux-cluster/intro/index.html*
- Informationen zu Umgebungen auf dem LRZ-Linux-Cluster erhalten Sie unter *http://www.lrz-muenchen.de/services/compute/linux-cluster/develop/index.html*

Viel Erfolg beim Bearbeiten!

Die Abgabe ist bis 03.11.2009, 9.00 Uhr im Raum MI 2.5.59 möglich.

Alle Programme finden Sie zum Herunterladen auf der Praktikumsseite unter:
`http://www5.in.tum.de/wiki/index.php/Masterpraktikum_Scientific_Computing_-_High_Performance_Computing_-_Winter_09`