Karlsruhe Institute of Technology (KIT) WS 2016/2017
Institute of Computer Science and Engineering (ITEC)
Chair for Computer Architecture and Parallel Processing (CAPP)

Prof. Dr. Wolfgang Karl
Prof. Dr. Götz Alefeld
M. Sc. Thomas Becker
Dipl.-Math. Markus Hoffmann

Softwarepraktikum Parallele Numerik

Projekt 1 — Poisson-Gleichung und OpenMP

OpenMP und Testtools

Aufgabe 1:
Diese Aufgabe wird Stück für Stück in ein Programmierparadigma[1] einführen, dass Ihnen hilft, die im Folgenden aufgeführten Parallelisierungsmethoden umzusetzen.

a) Schreiben Sie ein Programm, welches für jeden Thread „Hello World, this is Thread", sowie die zugehörige Thread-ID ausgibt. Mittels

```
export OMP_NUM_THREADS=<anzahl>
```

kann man die Anzahl der Threads unabhängig von der Anzahl an Kernen setzen. Führen Sie das Programm mit 1, 4 und 8 Threads aus. Was können Sie bei der Reihenfolge der Meldungen beobachten?

b) In dieser Aufgabe soll die Kreiszahl π mittels numerischer Integration berechnet werden. Es ist bekannt, dass π folgenderweise berechnet werden kann: $\pi = \int_0^1 \frac{4.0}{(1+x^2)}dx$. Eine Approximation des Integrals ist $\sum_{i=0}^{N} F(x_i)\Delta x$, wobei $F(x_i) = \frac{4.0}{1+x^2}$ ist. Parallelisieren Sie das gegebene Programm `pi.c` mittels OpenMP.
Bei der OpenMP-Version soll die Aufsummierung ein Mal manuell (mittels Master, Critical oder Atomic-Konstrukt) und ein Mal automatisch (mittels Reduction) durchgeführt werden. Führen Sie das Programm auch mit mehr

[1] Die aktuelle Spezifikation von OpenMP finden Sie auf `www.openmp.org`. Diese Spezifikation enthält zu jeder Direktive ein Beispiel. Ein Tutorial zu OpenMP ist unter `http://www.llnl.gov/computing/tutorials/openMP/` zu finden.

und weniger Threads als Kernen aus, sowie mit unterschiedlichen Potenzen von 10 für N, und berechnen Sie die jeweilige Beschleunigung. Was beobachten Sie jeweils und was glauben Sie, ist die Ursache hierfür?

c) Die Mandelbrot-Menge oder auch das Mandelbrot-Fraktal ist die Menge aller komplexen Zahlen $c = x + iy, x \in [-2, 2), y \in [-2, 2)$, für welche die rekursiv definierte Folge $z_0, z_1, z_2, ...$ mit der Berechnungsvorschrift $z_{n+1} = z_n^2 + c$ und der Anfangsbedingung $z_0 = 0$ beschränkt bleibt.
Parallelisieren Sie das gegebene Programm `mandelbrot.c` mittels OpenMP. Verwenden Sie verschiedene Auflösungen und berechnen Sie jeweils die Beschleunigung der parallelisierten Version. Unterscheiden sich die Werte für die erreichte Beschleunigung von Aufgabe 1b? Falls ja, was denken Sie, ist die Ursache hierfür?

Aufgabe 2:
Um zu testen, ob die parallelisierte Version eines Programms effizient und korrekt[2] arbeitet, können einige Testwerkzeuge[3] beziehungsweise Arbeitstechniken eingesetzt werden. Sinn dieser Aufgabe ist es, einige dieser Testwerkzeuge und Arbeitstechniken für OpenMP kennen zu lernen.

a) Gegeben sei die Datei `demo_with_bugs.c`, welche mehrere Beispiele enthält, die in bestimmten Situationen zu Problemen führen können. Analysieren Sie die einzelnen Beispiele mit Hilfe des Intel Thread Checkers. Welche Probleme treten jeweils auf und wie können sie behoben werden?

b) Erstellen Sie ein Programm zur Multiplikation zweier quadratischer Matrizen mit zufälligen Gleitkommawerten einfacher Genauigkeit. Wie sollte bei der Berechnung im Hinblick auf die Cache-Ausnutzung vorgegangen werden, wenn die Matrizen zeilenweise oder spaltenweise im Speicher abgelegt werden? Parallelisieren Sie die Multiplikation und versuchen Sie sie anschließend mit verschiedenen Ansätzen, z.B. mit den in der Einführung vorgestellten Möglichkeiten, weiter zu beschleunigen. Überprüfen Sie die Auswirkungen jedes Optimierungsschritts auf die Laufzeit mit dem GCC- und dem ICC-

[2] Korrekt im Sinne des korrekten Daten- und Kontrollflusses, nicht im Sinne des richtigen Ergebnisses!

[3] Intel Thread Checker - Quick Reference Guide:
`http://software.intel.com/en-us/articles/intel-thread-checker-documentation/`
Intel MKL documentation:
`http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/`
ompp Usage Guide:
`http://projekt17.pub.lab.nm.ifi.lmu.de/ompp/usage.pdf`
Weiter Informationen können Sie für die meisten Programme mittels `man $tool` abrufen.
Intel Thread Checker ist auf den Rechnern: `i82pc29, i82pc30, i82pc31` verfügbar.
ompp ist auf `i82sn02, i82sn06` installiert und unter `/opt/ompp/bin` verfügbar

Compiler.
Welche Ihrer getesteten Maßnahmen erzeugen eine erkennbare Beschleunigung? Hat die Wahl des Compilers einen Einfluß auf den Erfolg der einzelnen Schritte? Betrachten Sie außerdem den Beschleunigungsfaktor bei unterschiedlichen Matrizengrößen.
Vergleichen Sie im letzten Schritt Ihre beste Implementierung mit der Version der Intel Math Kernel Library.

c) Untersuchen Sie zuerst mit Hilfe des OpenMP-Profilers `ompp` Ihre Mandelbrot-Anwendung aus Aufgabe 1c. Welches Problem können Sie in der Analyse erkennen? Erweitern sie anschließend die OpenMP-Anweisung um eine `schedule`-Klausel und testen Sie die unterschiedlichen Zuteilungsmethoden. Welche Beschleunigung lässt sich durch statisches, welche durch dynamisches Scheduling erreichen? Variieren Sie neben der *chunk size* auch die Auflösung des Mandelbrot-Fraktal, sowie dessen Iterationstiefe.

Parallelisierung

Aufgabe 3:
Im Rahmen der Parallelisierung ist es zunächst notwendig, einige Grundbegriffe wie Kennzahlen, Fehlerquellen und verfügbare Hardwarekomponenten zu kennen.

a) Was versteht man unter den Begriffen "Speedup" und "Efficiency" im Rahmen der Parallelisierung? Finden und erläutern Sie weitere Kennzahlen[4] aus dem Umfeld der Parallelverarbeitung.

b) Was sind Race Conditions? Wie entstehen sie und wie kann man diese vermeiden? Welchen Einfluss haben diese auf die Korrektheit eines Ergebnisses? Nennen Sie weitere, speziell im Rahmen der Parallelisierung auftretende Fehlerquellen beim Softwareentwurf.

c) Vergleichen[5] Sie Beschleuniger wie (GP)GPUs, CPUs, FPGAs und MICs auf Basis ihrer Eignung zur Verfahrensparallelisierung, Energieeffizienz und Anwenderfreundlichkeit. Warum liegt der Fokus dieses Praktikums auf CPUs und GPUs?

[4] Zur Erfüllung dieser Aufgabe sind **maximal** 5 weitere Maße aufzulisten.
[5] Vergleichen Sie tabellarisch **maximal** 2 Unterpunkte der genannten Bereiche für **maximal** 4 Beschleuniger. Wählen Sie dabei die Unterpunkte und Beschleuniger, die Ihnen am relevantesten erscheinen.

Aufgabe 4:
Ein im Rahmen der Methode der Finiten Elemente benötigtes Lösungsverfahren ist das Gauß-Seidel-Verfahren. Der grundlegende Lösungsalgorithmus zur iterativen Lösung linearer Gleichungssysteme der Form $Au = h^2 f,\ A \in \mathbb{R}^{n\times n}$ ist hier gegeben durch

Wähle Startvektor $u^0 \in \mathbb{R}^n$
for $k = 0, 1, \dots$
  for $j = 1, 2, \dots, n$

$$u_j^{k+1} = \frac{1}{a_{j,j}}(h^2 f_j - \sum_{i=1}^{j-1} a_{j,i} u_i^{k+1} - \sum_{i=j+1}^{n} a_{j,i} u_i^k)$$

a) Gegeben sei das Gleichungssystem $Au = h^2 f$ in der Form

$$\begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix} \begin{pmatrix} u_{1,1} \\ \\ \vdots \\ \\ u_{n,n} \end{pmatrix} = h^2 \begin{pmatrix} f(x_1, y_1) \\ \\ \vdots \\ \\ f(x_n, y_n) \end{pmatrix} \quad \text{mit } T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}$$

sowie der Einheitsmatrix I und $u_{i,j} = u(x_i, y_j)$. Zudem seien gegeben:

- Die rechte Seite $f(x, y) = 32(x(1-x) + y(1-y)),\ (x, y) \in (0, 1)^2$,
- Die Randbedingung $u(0, y) = u(x, 0) = u(1, y) = u(x, 1) = 0$ sowie den Matrixdimensionen $T \in \mathbb{R}^{n\times n}$, $A \in \mathbb{R}^{n^2\times n^2}$, $n = \frac{1}{h} - 1$,
- Die Diskretisierungsweite $(x_{i+1}, y_{j+1}) = (x_i + h, y_j + h)$,
- Die Verfeinerung $h = \frac{1}{2^l}$ mit $l \subset \mathbb{N}$.

Programmieren Sie eine serielle Version des Gauß-Seidel-Verfahrens mit geeigneter Abbruchbedingung und messen Sie die Laufzeiten bei verschiedenen Verfeinerungen. Achten Sie auf die Korrektheit[6] ihrer Ergebnisse und nutzen Sie die dünne Struktur von A.

[6] Für die gebene rechte Seite ergibt sich eine analytische Lösung der Form $u(x, y) = 16x(1-x)y(1-y) \quad \forall (x, y) \in (0, 1)^2$.

b) Entwickeln Sie eine naive parallele Variante des Verfahrens. Untersuchen Sie diese Implementierung mit Hilfe der bereits kennen gelernten Tools. Was ist dabei festzustellen und warum?

c) Da das Gauß-Seidel-Verfahren Datenabhängigkeiten innerhalb der einzelnen Iterationsschritte zeigt, eignet sich eine naive Herangehensweise nicht, um dieses Verfahren ausreichend zu parallelisieren. Ihre Aufgabe ist es, eine sinnvolle Methodik zu entwickeln, mit welcher sich dieses Verfahren parallelisieren lässt. Messen Sie Speedup und Efficiency der Implementierung ihrer Methodik unter den oben gewählten Verfeinerungen und bewerten Sie damit die Qualität Ihrer Lösung. Testen Sie die Implementierung zusätzlich auf Skalierbarkeit.

Partielle Differentialgleichungen

Aufgabe 5:
In dieser Aufgabe werden Sie Schritt für Schritt ein Lösungsverfahren zu einer gegebenen partiellen Differentialgleichung implementieren. Zum Einsatz kommt dabei die Methode der Finiten Elemente in ihrer einfachsten Form, der Finiten Differenzenmethode.
Als erstes Modellproblem soll die Poisson-Gleichung mit homogenen Dirichletschen Randbedingungen betrachtet werden:

$$\begin{aligned} -\Delta u(x,y) &= f(x,y), \quad && (x,y) \in \Omega = (0,1)^2 \\ u(x,y) &= 0 && (x,y) \in \Gamma. \end{aligned} \tag{1}$$

a) Welche Bedingungen müssen an u, f, Ω und Γ gestellt werden, damit das Modellproblem (1) korrekt gestellt ist?

b) Gesucht ist f mit:

$u(x,y) = \sin(2M\pi x)\sin(2N\pi y)$ erfüllt (1) für $M, N \in \mathbb{N}$, $(x,y) \in \Omega$.

Geben Sie dieses f an.

c) Lösen Sie die gegebene Partielle Differentialgleichung[7] mit Hilfe der aus der-Einführung bekannten Methode der Finiten Differenzen für ein in jede Raumrichtung uniformes h in verschiedenen Abstufungen. Als Lösungsmethodik des entstehenden Gleichungssystems soll das parallele Gauß-Seidel-Verfahren zum Einsatz kommen, welches Sie bereits entworfen haben. Stellen Sie die

[7] Für die gegebene rechte Seite sind M und N im Rahmen der angegeben Voraussetzungen frei wählbar.

erhaltenen Lösungen anschaulich dar. Handelt es sich bei dieser Lösungsmethodik um eine h-, p- oder hp-FEM? Begründen Sie ihre Antwort.

Aufgabe 6:
Im Rahmen des Lösungsverfahrens für partielle Differentialgleichungen mit der Methode der Finiten Elemente kommen häufig Krylow-Unterraumverfahrens zum Einsatz, um das resultierende lineare Gleichungssystem zu lösen, da diese für die resultierenden Matrixeigenschaften bessere Konvergenzgeschwindigkeiten zeigen, als andere iterative Verfahren.

a) Stellen Sie zunächst eine Liste mit häufig verwendeten Krylow-Unterraumverfahren zusammen und wählen Sie dann anhand der Eigenschaften und Voraussetzungen der Verfahren einen geeigneten[8] Kandidaten aus, um das aus (1) resultierende Gleichungssystem zu lösen. Entwickeln Sie zunächst eine serielle Variante dieses Verfahrens, die zum Testen der Korrektheit ihrer Ergebnisse herangezogen werden kann. Parallelisieren Sie diese nun mit Hilfe von OpenMP unter Beachtung verschiedener Granularitätsstufen, prüfen Sie die Ergebnisse auf Korrektheit und analysieren Sie ihre Variante mit Hilfe der kennengelernten Werkzeuge.

b) Binden Sie die eben entwickelte parallele Version des Krylow-Unterraumverfahrens in ihr Programm aus Aufgabe 5 ein. Bestimmen Sie Speedup und Efficiency des Gesamtprogramms gegenüber der Version aus Aufgabe 5c. Warum ist der Einsatz eines Vorkonditionierers in Gesamtzusammenhang dieser Aufgabenstellung nicht sinnvoll? Unter welchen Voraussetzungen ändert sich das?

c) Welche Softwarebibliotheken zur Anwendung der Methode der Finiten Elemente können Sie im Internet finden? Worin liegen die Gemeinsamkeiten, worin unterscheiden sich diese? Welche dieser Bibliotheken würde sich am ehesten für die Verwendung in diesem Praktikum eignen und warum?

Version: 24. Oktober 2016

[8]Hinweis: Sie dürfen diverse Methoden zur Vorkonditionierung außer Acht lassen. Verwenden Sie die Grundform des entsprechenden Verfahrens.