Praktikum 4: OpenMP Parallelisierung serieller Programme

M.Thaler, 2/2016, ZHAW

1 Einführung

In diesem Praktikum werden sie zwei serielle Programme mit Hilfe von OpenMP parallelisieren. Zum einen ein Heat-Diffusion Programm zur Bestimmung der Temperaturverteilung auf einem 2D Mesh mit gegebenen Randwerten, sowie eine automatische Graustufen-Anpassung von Graustufenbidlern. sie werden kennenlernen, dass die Parallelisierung von seriellen Programmen je nach Problemstellung relativ einfache, aber auch aufwendig sein kann.

2 Heat Diffusion

Im Verzeichnis `serialpar/a1` finden sie eine einfache serielle Implementation zur Lösung der Heat Diffusion Differentialgleichung (siehe auch Algorithmusbeschreibung "Heat Diffusion") mit einer fixen Anzahl von Schritten.

Hinweis: das Programm verwendet für die graphische Ausgabe das netpbm-Format und für die Visualisierung `eog` (eine Gnome-Applikation). Falls sie `eog` nicht zur Verfügung haben, passen sie bitte im Modul `display.c` die Ausgabe entsprechend an, z.B. Daten in ein File schreiben.

2.1 Aufgabe 1

Parallelisieren sie das serielle Heat-Diffusion Programm mit Hilfe von OpenMP. Wie kann dieses 2-dimensionale mesh-basierte Problem parallelisiert werden? Implementieren sie verschiedene Möglichkeiten und bestimmen sie jeweils, wie OpenMP das Problem auf Threads abbildet und welche Performance (Speedup und Effizienz) jeweils erreicht werden kann.

Hinweise: für die Performance-Messungen dürfen keine `printf()`'s, etc. aufgerufen werden (auskommentieren).

2.2 Aufgabe 2

Erweitern sie das Heat-Diffusion Programm um ein Abbruchkriterium (keine fixe Anzahl von Iterationen wie in Aufgabe 1). Das Programm soll abbrechen, wenn gilt: $\Delta U(x,y) < \epsilon \; \forall x,y$. Wählen sie für ϵ die Werte $10^{-3}, 10^{-4}, 10^{-6}$. Vergleichen sie die Performance Resultate mit den denjenigen aus Aufgabe 1.

3 Histogram Equalization

Im Verzeichnis `serialpar/a2` finden sie eine serielle Implementation der Histogramm Equalization, wo im Zielbild eine gleichmässige Graustufenverteilung, d.h. ein flaches Histogramm, angestrebt wird. Der Algorithmus ist relativ einfach und besteht aus folgenden Schritten:

1. erstellen eines Histogramms des Quellenbildes
2. daraus das kumulative Histogramm berechnen
3. das kumulative Histogramm auf den Datenbereich $[0...255]$ normieren

4. normiertes kumulatives Histogramm als Lookup-Tabelle für die Grauwerttransformation des Quellenbildes verwenden:
    - Graustufe von Pixel $P(x, y)$ im Quellenbild $\rightarrow$ Index in kumulativem Histogramm
    - Wert bei Index $\rightarrow$ Graustufe von Pixel $P(x, y)$ im Zielbild

3.1 Aufgabe

Parallelisieren sie die serielle Implementation der Histogramm Equalization mit OpenMP so, dass das resultierende Programm mit der Anzahl Threads soweit als möglich skaliert (d.h. unabhängig von einer fixen Anzahl Threads ist). Überlegen sie sich, wie Daten und Algorithmus angepasst werden können, um die geforderte Parallelisierung möglichst optimal zu erreichen.

Es stehen zwei Bilder zur Verfügung (`london.pgm` und `shelf.pgm`). Der Filename des Bildes kann auf der Kommandozeile an `main.c` übergeben werden, defaultmässig wird `london.pgm` eingelesen.

Hinweis: beachten sie, dass das Bild in einen 1D Array eingelesen wird.

4 Take Home

Was ist bei der Parallelisierung von seriellen Programmen mit OpenMP zu beachten bezüglich Datenstrukturen und Algorithmen zu beachten?

Notieren sie sich die wichtigsten Punkte, die sie im Zusammenhang mit diesem Praktikum kennengelernt haben.