Wissenschaftliches Rechnen III / CP III
Übungsblatt 3

Tizia Kaplan (545978)
Benjamin Dummer (532716)

18.05.2016

Online-Version: `http://www.github.com/BeDummer/CP3_UE3`

Aufgabe 3.1

Die entsprechenden Funktionen wurden in der gegebenen Datei `cg.c` ergänzt. Zur Überprüfung der implementierten Funktion `laplace_2d` wurde die Matlab-Routine `testlaplace.m` genutzt. Hier wurde der „zufällige "(aber immer gleiche) Vektor des C-Programms in die Matlab-Routine kopiert und die beiden Ergebnisse wurden verglichen. Es war festzustellen, dass die implementierte Funktion korrekt arbeitet. Ähnlich wurde für die Verifizierung der gesamten Methode vorgegangen. Hierfür wurde die Matlab-Routine `ue3_verify.m` genutzt. Wobei auch hier der Vektor des C-Programms in die Matlab-Routine kopiert wurde und die beiden Ergebnisse verglichen wurden. Auch die Methode der konjugierten Gradienten konnte korrekt implementiert werden.

Aufgabe 3.2

In der beigefügten Datei `cg.cu` wurden die Kernel-Funktionen `laplace_2d_gpu` und `vec_add_gpu` für die (Host-)Funktionen `laplace_2d` und `vec_add` implementiert und eine Berechnung der Laufzeit durchgeführt.

Die Einteilung der Blockstruktur wurde in 2 Varianten untersucht:

a) 8x8-**Blöcke**: Das Programm bildet Blöcke im Format 8x8 und ordnet diese je nach Gesamtgröße des Gitters in einem quadratischen Grid an, dadurch ist die Größe des Grids auf Vielfache von 8 beschränkt.

b) 32x32-**Blöcke**: Das Programm bildet Blöcke im Format 32x32 und ordnet diese je nach Gesamtgröße des Gitters in einem quadratischen Grid an, dadurch ist die Größe des Grids auf Vielfache von 32 beschränkt.

In den folgenden Tabellen ist der berechnete *Speedup* aufgeführt. Aufgrund von Schwankungen zwischen verschiedenen Durchläufen des Programms wurden die Ergebnisse von jeweils 6 Programmdurchläufe mit denselben Parametern gemittelt. (Zur Übersichtlichkeit wurde auf die Angabe der Standardabweichung verzichtet. Die Schwankungen hielten sich in einem geringen Rahmen.)

Speedup der zwei Funktionen `laplace_2d` und `vec_add` für die Ausführung auf der GPU vs. CPU mit 8x8 Blöcken

$N_x + 2 = N_y + 2$	8	16	32	64	128	256	512
laplace_2d	0.03	0.1	0.46	1.71	0.92	3.62	9.31
vec_add	0.05	0.07	0.17	0.68	2.25	4.32	6.55

Speedup der zwei Funktionen `laplace_2d` und `vec_add` für die Ausführung auf der GPU vs. CPU mit 32x32 Blöcken

$N_x + 2 = N_y + 2$	32	64	128	256	512
laplace_2d	0.43	1.67	0.92	3.62	10.5
vec_add	0.16	0.65	2.40	4.82	7.6

Es gibt nur geringfügige Unterschiede beim Speedup zwischen den 2 Blockgrößen, welche im Rahmen der statistischen Schwankungen liegen. Auffällig ist, dass sich für die Funktion `vec_add` ein monotoner Anstieg des Speedups mit der Gittergröße ergibt, wobei für die Funktion `laplace_2d` ein lokales Minimum bei $N_x + 2 = N_y + 2 = 128$ in der Messreihe existiert. Hier scheinen verschiedene Komponenten zu konkurrieren, eine Analyse mit `nvprof` könnte genauere Hinweise liefern.

Anhänge

- Datei: `cg.cu` (Hauptprogramm)
- Datei: `testlaplace.m` (Verifizierung der implementierten Funktion `laplace_2d` mit der vorhandenen Matlab-Routine)
- Datei: `ue3_verify.m` (Verifizierung der implementierten Methode der konjugierten Gradienten mit der vorhandenen Matlab-Routine)