Übungen zur Vorlesung "Architektur und Programmierung von Grafik- und Koprozessoren"
Übungsblatt 9

Sommersemester 2018

9 Programmieren mit CUDA

Aufgabe 9.1

Ihre Aufgabe ist es, ein bereits korrekt ausführendes CUDA Programm zu optimieren. Ermitteln Sie mit `nvcc` die Anzahl an Registern, die der Compute Kernel des beigefügten, parallelen Programms benötigt. Der `nvcc` Compiler ist auf dem Testsystem unter `/usr/local/cuda/bin` installiert. Bestimmen Sie mit dem Occupancy Calculator Excel Spreadsheet [1], das Sie unter `/usr/local/cuda/tools` finden können, eine optimale Einteilung der Berechnungen auf *Thread Blöcke*. Berücksichtigen Sie dabei die Compute Capability der GPU, für die Sie optimieren (im Testsystem ist etwa eine NVIDIA Titan V Grafikkarte verbaut) und dokumentieren Sie diese als Teil Ihrer Lösung. Passen Sie das Programm entsprechend an. Vergleichen Sie die Ausführungszeiten Ihres Programms mit der unoptimierten Variante. Geben Sie die Ausführungszeiten für Thread Blöcke der Größen 2^3, 2^4, ... bis einschließlich 2^{10} an und plotten Sie diese (achten Sie darauf, dass dabei das Programm weiterhin korrekt ausgeführt wird). Tragen Sie in das Diagramm auch die *GPU Auslastung* bzgl. des Kernels für die jeweilige Blockgröße ein. (5 Punkte)

Aufgabe 9.2

Erklären Sie, was das Threading Modell neuerer NVIDIA GPUs vom Multi-Threading Modell moderner CPUs unterscheidet. Strukturieren Sie Ihre Antwort gemäß der nachfolgenden Überlegungen:

1. Welche Funktionseinheit (Hardware oder Software) ist für das Scheduling von Threads auf der jeweiligen Architektur zuständig?
2. Werden eventuell Threads synchron ("lockstep") bearbeitet? Wenn ja, auf welcher Architektur? Welche Konsequenzen hat das für den Programmablauf? Wie vergleicht sich dieses Ausführungsmodell mit SIMD Architekturen?
3. GPUs verwalten viele Threads gleichzeitig. Häufig sind zur selben Zeit mehr Thread Gruppen aktiv, als es Streaming Multiprozessoren gibt, sodass die Scheduling Einheiten den Gruppen von Threads wechselnd Zeitfenster für ihre Berechnungen zuweisen müssen. Unter welchen Gesichtspunkten wird zwischen den Thread-Ausführungen hin und her geschaltet?
4. Das Umschalten zwischen Gruppen von Threads geht auf GPUs besonders schnell. Warum ist das der Fall?

(5 Punkte)

[1] Oder mit dem Tool `https://xmartlabs.github.io/cuda-calculator/`

Aufgabe 9.3

a.)
Implementieren Sie Matrixmultiplikation zweier Matrizen A und B mit Dimensionen (# Zeilen $\times$ # Spalten) $N \times M$ sowie $M \times N$ mit CUDA. Dazu verwenden Sie das beigefügte Gerüstprogramm. Ihr paralleles GPU Programm verwendet zur Berechnung ein zweidimensionales Gitter der Größe $N \times N$, sodass jedem Matrixelement aus der Ergebnismatrix C ein einzelner Thread zugeordnet wird. Jeder Thread berechnet das ihm über die 2D Thread ID zugeordnete Skalarprodukt aus Zeilen- und Spaltenvektor. Das Gitter teilt sich in Blöcke der Größe 32×32 auf. Sie dürfen vereinfachend annehmen, dass N und M ganzzahlige Vielfache von 32 sind. Die Berechnungen führen Sie direkt im *Globalen Speicher* der Grafikkarte aus. (5 Punkte)

b.)
Implementieren Sie eine optimierte Variante der Matrixmultiplikation. Es gelten die gleichen Rahmenbedingungen wie aus Aufgabenteil **a.)**. Nun optimieren Sie Ihr Programm, indem Sie das Skalarprodukt aus Zeilen- und Spaltenvektor im *Shared Memory* durchführen. Bei der Berechnung des Skalarprodukts iterieren Sie über Blöcke der Größe 32×32 und schreiben die zugehörigen Matrixelemente in ein Shared Memory Array. Da die Blockgröße im Gitter ebenfalls 32×32 beträgt, ist jeder Thread im Block für ein Matrixelement zuständig. Führen Sie nun *stückweise* Skalarprodukte innerhalb der Blöcke durch. Sobald Sie einen Block abgearbeitet haben, iterieren Sie "nach rechts" (Matrix A) bzw. "nach unten" (Matrix B). Dabei akkumulieren Sie die Ergebnisse der lokalen Skalarprodukte und weisen die Summe zum Schluss dem Matrixelement in der Ergebnismatrix C zu, für das der jeweilige Thread zuständig ist. Bedenken Sie, dass Threads, die Operationen im Shared Memory durchführen, synchronisiert werden müssen. (5 Punkte)

Abgabe bitte bis zum 11.07.2018, 22:00h in Ilias.