Arbeitsblatt OpenCL

Laden Sie das xx_WS_OpenCL.zip vom AD [1] herunter und importieren Sie es in Ihren Eclipse Workspace. Für die Ausführung auf x86 CPUs sowie ATI Grafikkarten benötigen Sie das AMD APP SDK [2] und für die Ausführung auf nvidia Karten lediglich einen aktuellen Treiber.

Aufgabe 1
In dieser Aufgabe werden Sie die Berechnung des Mandelbrot Sets, welches Sie bereits zu Beginn des Semesters parallelisiert haben, mit Aparapi [3] für OpenCL umsetzen. Sie finden die Struktur dafür im Package *ch.fhnw.conpr.mandel* in welchem auch bereits eine Sequentielle sowie parallele Java Implementierung vorhanden ist. Zudem existiert ein Source Folder mit Unit Tests, mit welchen Sie Ihre Implementierung überprüfen können.

Aufgaben:

1. Führen Sie zunähchsts die Unit Tests aus. Welche Geräte werden auf Ihrer Umgebung unterstützt?

2. Implementieren Sie nun den MandelbrotKernel in der Klasse *ch.fhnw.conpr.mandel.AparapiMandelbrotCalculator*. Sie müssen dazu lediglich die Funktion *run* implementieren.

3. Führen Sie nun die Klasse *ch.fhnw.conpr.mandel.Mandelbrot* aus. Sind sie mit der Performance zufrieden? Wie könnten Sie die Performance auf der GPU weiter steigern?

Tipp: Spielen Sie mit der Work Group Size in der Funktion *calculateMandelbrotIterations*

Aufgabe 2
Nachdem Sie in der ersten Aufgabe den Einfluss der Wahl der Work Group Size auf die Performance gesehen haben. Können Sie in dieser Aufgabe mit unterschiedlichen Arbeitsverteilungen auf die Work Items experimentieren.

Die Aufgabe des Kernels ist es eine Strahlendosis innerhalb eines CT-Grids zu berechnen, wobei die Dosis abhängig von der Dichte des Gewebes an einer Position innerhalb des Grids ist. Für diesen Test gilt:

$$dosis_{xyz} = \sum_{i=0}^{density_{xyz}} \exp(i)$$

Die Last eines Kernels wird also durch den Integer Wert im *densities* Array definiert. Beim Ausführen der Klasse AparapiGridCalculation wird die Ebene in der Mitte des Grids auf die Konsole ausgegeben. So haben Sie die Möglichkeit ein Gefühl für die Dichtefunktionen zu bekommen. Am Ende der Ausführung wird de Laufzeit für den Java Thread Pool (JTP), die CPU und GPU ausgegeben.

Aufgaben:

1. Führen Sie nun die Klasse AparapiGridCalculation wiederholt aus und ersetzen Sie die Dichtefunktion zwischen den Aufrufen in der Funktion *densityFunction*. Notieren Sie sich die Laufzeiten und wie die Dichte im Grid verteilt ist.

2. Wie erklären Sie sich die Unterschiede in der Laufzeit? Wann erreicht die GPU den grössten Performance Gewinn, wann den schlechtesten?

[1] E1862_4Ia\conpr\xx_OpenCL\Lecture\WS_OpenCL.zip
[2] http://developer.amd.com/tools-and-sdks/heterogeneous-computing/amd-accelerated-parallel-processing-app-sdk/
[3] https://code.google.com/p/aparapi/