Praktikum 8: OpenCL: Monte Carlo and Heat Diffusion

M.Thaler, 2/2016, ZHAW

1 Einführung

In diesem Praktum werden wir uns mit zwei Anwendungen beschäftigen, die typischerweise sehr effizient auf GPU's implementiert werden können: Monte Carlo Verfahren (statisches Verfahren) und numerisches Lösen von mehrdimensionalen Differntialgleichungen (Beispiel Heat Diffusion).

2 Monte Carlo Verfahren

Monte Carlo Verfahren eignen sich überall dort, wo Systeme analysiert werden müssen, die nicht analytisch lösbar sind, sehr aufwendige Berechnungen erfordern oder deren Parameterwerte statistisch verteilt sind (z.B. Toleranzen von elektronischen Bauteilen, etc.).

2.1 Mein Kollege der Marketing Manager

Ein Kollege von ihnen ist Marketing Manager und gerade daran ein neu entwickeltes Produkt auf den Markt bringen. Das Upper Management velangt nun von ihm eine Risko-Abschätzung für den Netto-Profit im 1. Jahr.

Der Netto-Profit berechnet sich wie folgt:

Netto-Profit = Verkaufsvolumen * (Verkaufspreis - Stückkosten) - Fixkosten.

Für das Risiko-Modell gibt es drei Marktszenarien, die alle gleich wahrscheinlich sind:

1. **SLOW**: Verkaufsvolumen 50'000 Stück, Verkaufspreis Fr. 11.-
2. **OK**: Verkaufsvolumen 75'000 Stück, Verkaufspreis Fr. 10.-
3. **HOT**: Verkaufsvolumen 100'000 Stück, Verkaufspreis Fr. 8.-

Die Stückkosten liegen im Bereich Fr. 5.50.- und Fr. 7.50.-, mit grosser Wahrscheinlichkeit bei Fr. 6.50.-, und die Fixkosten betragen Fr. 120'000.-

Eine einfache Schätzung basierend auf Mittelwerten ergibt einen Profit von 117'750.- im ersten Jahr. Die Frage ist, ob diese einfache Schätzung zu optimistisch oder zu pessimistisch ist?

2.2 Aufgabe 1

Implementieren Sie eine Monte Carlo Simuation für das Risko-Modell ihres Kollegen. Jedes Work Item implementiert einen möglichen Fall des Risiko-Modells (ein zufallsbasierter Netto-Profit) und wählt dazu eines der Marktszenarien mit Hilfe eines gleichverteilten Zufallsgenerators, den Stückpreis mit Hilfe einer Dreickecksverteilung. Die Schätzung des Netto-Profits besteht dann aus dem Mittelwert aller Resultate der Work Items.

Im Verzeichnis `oclApps/MonteCarlo/a1` finden sie das Hauptprogramm `main.c` und das OpenCL File `profit.cl` mit den Kerneln "profit" und "reduction", die sie ergänzen bzw. implementiern müssen. Dem Kernel "profit" kann die Variable `iterations` übergeben werden, ignorieren sie diese vorerst.

In `profit.cl` haben wir die beiden Zufallsgeneratoren vorbereitet: eine Gleichverteilung für die Wahl des Risiko-Models (Rückgabewert modulo 3) und die Dreiecksverteilung für die Stückkosten (Angabe von Minimum, wahrscheinlichstem Wert, Maximum). Als Parameter können sie beim Hauptprogram die Anzahl Work Items in K (1024) angeben.

Ist die Schätzung ihres Kollegen zu optimistisch oder zu pessimistisch?

2.3 Aufgabe 2

Erweitern sie nun den ”profit” Kernel so, dass er mehrere Risko-Modelle berechnet (Anzahl in `iterations`) und den Mittelwert dieser Berechnungen zurückgibt. Die Anzahl Iteration kann als zweiter Parameter an `./main.e` übergeben werden.

Bestimmen Sie die Rechenzeit für 1024*1024 (also 1048576) Zufallssimulationen, indem sie entweder `./main.e 1024 1` oder `./main.e 1 1024` aufrufen. Welche Variante ist schneller und wieso?

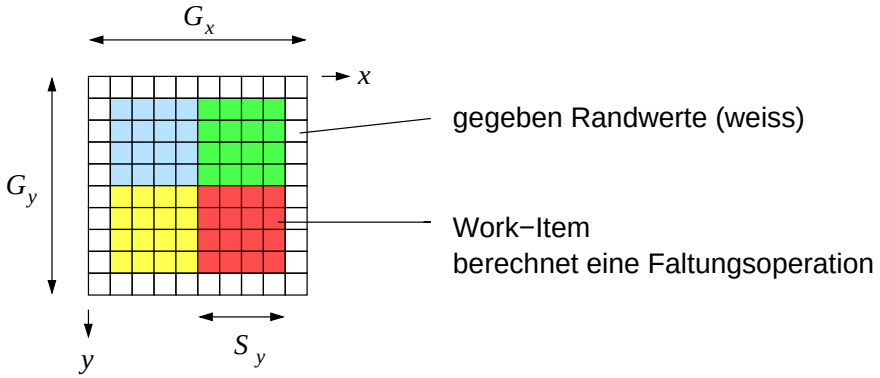
3 Heat Diffusion

Im Praktikum `serialPar` haben Sie mit OpenMP ein Programm zur Bestimmung der Temperaturverteilung auf einem 2-d Grid mit gegebenen Randwerten parallelisiert. In dieser Aufage werden wir die gleiche Anwendung auf der GPU implementieren und dabei den Umgang mit 2-dimensionalen Datenstrukturen kennen lernen, siehe dazu auch die Beschreibung *Heat Diffusion*.

3.1 Zweidimensionale Daten und Work-Groups

Die Temperaturverteilung zum Zeitpunkt T_{i+1} lässt sich mit Hife einer Faltung der Temperaturverteilung zum Zeitpunkt T_i mit einem 3×3 Faltungskern berechnen, d.h. in jedem Grid-Punkt muss eine *Faltungsoperation* mit dem Faltungskern ausgeführt werden.

In OpenCL wird diese Operation pro Datenpunkt durch ein Work-Item berechnet, die Work-Items können dabei parallel ausgeführt werden können. Die Work-Items selbst lassen sich hier zu 2-dimensionalen Work-Groups zusammenfassen. Unten stehende Figur zeigt einen 2-d Grid von Datenpunkten und farbig markiert vier Regionen, die jeweils einer Work-Group entsprechen:



Der Einfachheit halber gehen wir in diesem Praktikum davon aus, dass die Anzahl Gridpunte in x- und y-Richtung jeweils ein vielfaches der Seitenlänge einer Work-Group ist, z.B: $G_x = n \cdot S_x + 2$.

Für die Berechnung der Faltung muss ein OpenCL Kernel implementiert werden, der einen Faltungskern berechnet und wie folgt *gestartet* wird:

```
clEnqueueNDRangeKernel(queue,kern,dim,offset,wItems,wGroups,0,NULL,ev);
```

Die wesentlichen Paramter müssen wie folgt übergeben werden:

`dim = 2`	skalar	Anzahl Dimensionen
`offset[2]`	array	*Startkoordinaten* der Work-Group, z.B. gelb = {1, 5}.
`wItems[2]`	array	Anzahl Work-Items in jeder Dimension, Beispiel oben {8, 8}.
`wGroups[2]`	array	Grösse der Work-Items in jeder Dimension, Beispiel oben {4, 4}.

3.2 Aufgabe

Implementieren die Heat Diffusion mit OpenCL. Sie müssen dazu das Hauptprogramm im Verzeichnis `oclApps/heatDiffusion` ergänzen (Aufruf Kernel) und den Kernel implementieren. Beachten Sie bitte, dass der zweidiemensionale Grid in einem eindimensionalen Array der Länge $widthx * widthy$ gespeichert wird (zeilenweise), darum wird dem Kernel die Zeilenlänge (`widthX`) übergeben. Zufgriff auf das y-te Element in Zeile x berechnet sich dann wie folgt:

```
index_xy = x * widthX + y
```

Der neue Temperaturwert an der Position x, y berechnet sich wie folgt (Stencil):

$$u(x,y)_{n+1} = u(x,y)_n + K \cdot (u(x-1,y)_n + u(x+1,y)_n + u(x,y-1)_n + u(x,y+1)_n - 4 \cdot u(x,y)_n)$$

mit $K = 0.1$

4 Take Home

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

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