Übungsblatt 9

06.07.2018 / B. Leder

Wissenschaftliches Rechnen III / CP III

Aufgabe 9.1: *Monte-Carlo-Simulation: GPU Implementierung*

Implementieren Sie die Aktualiserung aller Spins (ein sweep) auf der GPU. Die Messung der Observablen nach jedem sweep (oder einer festen Anzahl) kann auf der CPU erfolgen. So ist für große Gitter ein speedup von > 10 möglich.

Hinweise

- Beachten Sie die Hinweise in Aufgabe 8.2.
- Schachbrett-Aufteilung: Für eine gerade Anzahl von Punkten in alle Richtungen ist eine mögliche Aufteilung der Gitterpunkte in zwei Untermengen eine Schachbrett-Aufteilung in schwarze und weiße Punkte. Man definiert die Summe der Koordinaten $s = \sum_{\mu=1}^{d} x_\mu$. Die weißen Punkte sind die, für die s gerade ist. Die schwarzen Punkte sind die, für die s ungerade ist. Überzeugen Sie sich, dass die weißen (bzw. schwarzen) Spins unabhängig voneinander aktualisiert werden können.
- Nächste-Nachbarn-Array: Das zweidimensionale Array der nächsten Nachbarn kann wie folgt auf das Device kopiert werden

```
int *d_nn;
CHECK(cudaMalloc((void**)&d_nn,nvol*(2*ndim+1)*sizeof(int)));
CHECK(cudaMemcpy(d_nn, nn[0], nvol*(2*ndim+1)*sizeof(int), cudaMemcpyHostToDevice));
```

- Konstanten können Sie als globale Variablen auf dem Device definieren. Zum Beispiel für λ und κ

```
// ausserhalb von Funktion/Kernel
__device__ double devLambda, devKappa;
...
// in einer Host-Funktion
CHECK(cudaMemcpyToSymbol(devLambda, &lambda, sizeof(double)));
CHECK(cudaMemcpyToSymbol(devKappa, &kappa, sizeof(double)));
```

`devLambda` und `devKappa` können dann in jedem Kernel verwendet werden.

Aufgaben:

1. Verifizieren Sie ihre Implementierung indem Sie den sweep über die zwei Untergruppen auch auf der CPU implementieren und die gleichen Zufallszahlen für einen sweep auf der CPU und auf der GPU mit der gleichen Ausgangskonfiguration der Spins benutzen. Das Ergebnis muss dann bis auf Rundungsfehler übereinstimmen.

2. Messen Sie den speedup für ein 32x32x32 Gitter.

15 Punkte