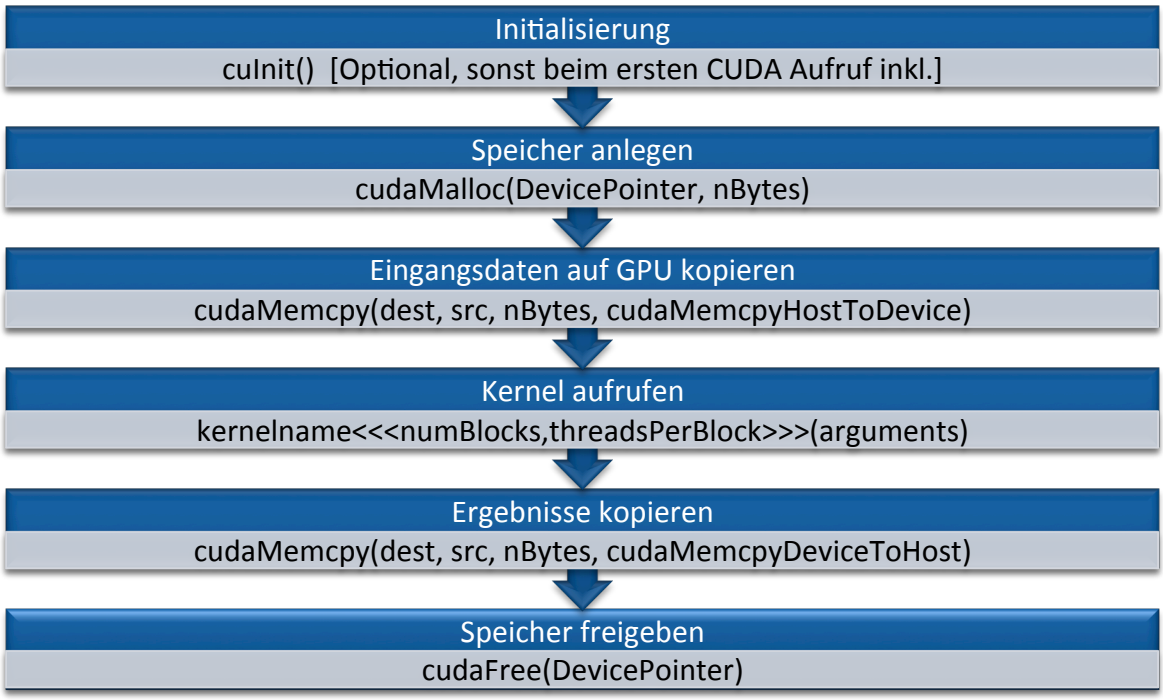
ETI – GPU Programmierung

CUDA Runtime Cheat Sheet

Programmablauf



Kernel-Tricks

Kernel Deklaration

- __global__ kernelname(args) - vom Host aufrufbar
- __device__ kernelname(args) - nur innerhalb GPU

Globaler Thread-Index

- int idx=blockIdx.x*blockDim.x+threadIdx.x;

Shared Memory (nur innerhalb eines Threadblocks)

- __shared__ float As[BLOCK_SIZE][BLOCK_SIZE];

Thread Synchronisation (nur innerhalb eines Threadblocks)

- __syncthreads();

Programmübersetzung

- Wichtig: CUDA Dateien immer mit Endung **.cu** benennen!
- Programm übersetzen mit „nvcc <dateiname.cu>“ (z.B. nvcc foo.cu)
- ./a.out

Aufgaben

1. Schreiben Sie ein Programm, dass einen Vektor „a_host“ vom Typ „int“ mit 1.000.000 Elementen anlegt. Initialisieren Sie die Elemente so, dass der Wert eines Elements gleich dem Index ist (a[i]=i).
2. Legen Sie die Vektoren a_device, b_device und b_host mit der gleichen Größe und dem gleichen Typ an. Die *_device Vektoren sollen dabei auf der GPU liegen. Übertragen Sie a_host zu a_device, kopieren ihn dort zu b_device und holen ihn nach b_host zurück. Überprüfen Sie, dass alle Elemente von a_host und b_host identisch sind. Vergessen Sie nicht alle allokierten Daten wieder freizugeben.
3. Alle CUDA Funktionen geben einen Fehlercode zurück, den man abfangen sollte. Dieser Rückgabewerde ist vom Typ „cudaError_t“. Fügen Sie für alle CUDA Aufrufe des Programms von Aufgabe 2 das Abfangen des Rückgabewerts hinzu. Überprüfen Sie den Wert immer sofort, ob er mit „cudaSuccess“ übereinstimmt. Liefern Sie, wenn dies nicht der Fall ist, eine Fehlermeldung. Überprüfen Sie, ob Ihre Fehlerabfrage funktioniert, indem Sie mehr Daten mit cudaMemcpy übertragen, als allokiert sind.
   Hinweise:
   - Sie können den Fehler vom Typ cudaError_t mittels der Routine „cudaGetErrorString(cudaError_t error)“ in einen lesbaren String umwandeln.
   - Die Compilermakros „__FILE__“ und „__LINE__“ liefern den Dateinamen der Quelldatei und die Zeilennummer des Befehls.
   - Überlegen Sie, ob Sie die Fehlerabfrage in ein Compilermakro abstrahieren können.
4. Schreiben Sie ein Program, dass drei Vektoren „a_host“, „b_host“ und „c_host“ vom Typ „int“ mit 1.000.000 Elementen anlegt. Initialisieren Sie die Elemente von „a_host“ und „b_host“ so, dass der Wert eines Elements gleich dem Index ist (a[i]=i). Kopieren Sie a und b auf das GPU device und führen dort für alle Elemente die Operation c=a+b durch. Holen Sie das Ergebnis c zurück und überprüfen Sie es auf Korrektheit
5. Sichern Sie Ihren Kernel aus Aufgabe 1 so ab, dass er für beliebige Blockgrößen funktioniert.
6. Portieren Sie nun die Matrix-Matrix-Multiplikation auf die GPU, indem Sie jeden Thread ein Element der Ergebnismatrix C berechnen lassen.

Protokoll (Abgabe bis 05.02.2014)

Fertigen Sie ein Praktikumsprotokoll an, welches die Ergebnisse ihrer Arbeit beinhaltet. Das Protokoll sollte mindestens folgende Informationen enthalten:

- Name, Matrikelnummer
- Beschreibung der Implementierung von Aufgabe 6
- Zeitmessung für unterschiedliche Blockgrößen
- Diskussion der Ergebnisse (Vergleich mit den sequentiellen und mit MPI erreichten Geschwindigkeiten)