

# Übung 3

## Computational Physics III

Matthias Plock (552335)

Paul Ledwon (561764)

18. Mai 2018

### Inhaltsverzeichnis

1 Aufgabe 1	1
2 Aufgabe 2	2

### 1 Aufgabe 1

Auf der GPU wird die Addition von Arrays der Dimension  $N = 1024^2$  für verschiedene execution configurations, daher die Einteilung der Threads in zweidimensionale Blöcke und Grids untersucht. Der untersuchte Parameter ist die Laufzeit der Vektoraddition. Die Dimension  $N$  wird als Produkt von vier Faktoren dargestellt, wobei die Faktoren die Aufteilung in Block- und Gridgröße bestimmen. Tendenziell scheinen Verteilungen mit großen Griddimensionen und kleinen Blockdimensionen die kürzesten Laufzeiten zu haben. Aus den 820 möglichen Kombinationen werden die 40 schnellsten execution configurations aufgelistet:

0.00016 seconds for	gridX: 2	gridY: 262144	threadX: 2	threadY: 1
0.00016 seconds for	gridX: 131072	gridY: 1	threadX: 2	threadY: 4
0.00016 seconds for	gridX: 1	gridY: 262144	threadX: 1	threadY: 4
0.00016 seconds for	gridX: 2	gridY: 65536	threadX: 8	threadY: 1
0.00016 seconds for	gridX: 262144	gridY: 1	threadX: 4	threadY: 1
0.000161 seconds for	gridX: 65536	gridY: 2	threadX: 8	threadY: 1
0.000161 seconds for	gridX: 65536	gridY: 4	threadX: 1	threadY: 4
0.000161 seconds for	gridX: 65536	gridY: 4	threadX: 4	threadY: 1
0.000161 seconds for	gridX: 131072	gridY: 4	threadX: 2	threadY: 1
0.000161 seconds for	gridX: 1	gridY: 524288	threadX: 2	threadY: 1
0.000161 seconds for	gridX: 1	gridY: 131072	threadX: 8	threadY: 1
0.000161 seconds for	gridX: 131072	gridY: 4	threadX: 1	threadY: 2
0.000161 seconds for	gridX: 524288	gridY: 1	threadX: 2	threadY: 1
0.000161 seconds for	gridX: 65536	gridY: 2	threadX: 4	threadY: 2
0.000161 seconds for	gridX: 1	gridY: 131072	threadX: 2	threadY: 4
0.000161 seconds for	gridX: 1	gridY: 131072	threadX: 1	threadY: 8
0.000161 seconds for	gridX: 131072	gridY: 2	threadX: 4	threadY: 1
0.000161 seconds for	gridX: 131072	gridY: 1	threadX: 8	threadY: 1
0.000161 seconds for	gridX: 65536	gridY: 8	threadX: 2	threadY: 1
0.000161 seconds for	gridX: 65536	gridY: 8	threadX: 1	threadY: 2
0.000161 seconds for	gridX: 1	gridY: 65536	threadX: 1	threadY: 16
0.000161 seconds for	gridX: 1	gridY: 65536	threadX: 2	threadY: 8
0.000161 seconds for	gridX: 1	gridY: 131072	threadX: 4	threadY: 2
0.000161 seconds for	gridX: 1	gridY: 65536	threadX: 8	threadY: 2
0.000161 seconds for	gridX: 262144	gridY: 1	threadX: 1	threadY: 4
0.000161 seconds for	gridX: 65536	gridY: 2	threadX: 1	threadY: 8
0.000161 seconds for	gridX: 524288	gridY: 1	threadX: 1	threadY: 2
0.000161 seconds for	gridX: 8	gridY: 65536	threadX: 2	threadY: 1
0.000161 seconds for	gridX: 8	gridY: 65536	threadX: 1	threadY: 2
0.000161 seconds for	gridX: 4	gridY: 131072	threadX: 2	threadY: 1

```

0.000161 seconds for gridX: 4  gridY: 131072  threadX: 1  threadY: 2
0.000161 seconds for gridX: 4  gridY: 65536  threadX: 4  threadY: 1
0.000161 seconds for gridX: 262144  gridY: 2  threadX: 1  threadY: 2
0.000161 seconds for gridX: 2  gridY: 262144  threadX: 1  threadY: 2
0.000161 seconds for gridX: 2  gridY: 131072  threadX: 4  threadY: 1
0.000161 seconds for gridX: 4  gridY: 65536  threadX: 1  threadY: 4
0.000161 seconds for gridX: 2  gridY: 65536  threadX: 4  threadY: 2
0.000161 seconds for gridX: 2  gridY: 65536  threadX: 2  threadY: 4
0.000161 seconds for gridX: 65536  gridY: 1  threadX: 2  threadY: 8
0.000161 seconds for gridX: 65536  gridY: 1  threadX: 16  threadY: 1

```

Mit einer Laufzeit von  $160 \mu\text{s}$  ist die Konfiguration mit  $2 \cdot 262\,144$  Blöcken und  $2 \cdot 1$  Threads am schnellsten. Die langsameren Konfigurationen waren alle bei unter  $700 \mu\text{s}$ .

## 2 Aufgabe 2

Der Speedup ist ein Maß dafür, wie sehr ein numerisches Problem von Parallelisierung profitiert. Für den Fall, dass man GPU und CPU vergleicht, ist der Speedup wie folgt definiert

$$S = \frac{T_{\text{GPU}}}{T_{\text{CPU}}},$$

wobei  $T$  jeweils die Laufzeit des Problems auf der GPU bzw CPU bezeichnet.

Um die Skalierung des Laplaceproduktes auf der GPU zu untersuchen, wird der Speedup gegenüber des Laplaceproduktes auf der CPU für verschiedene Gitterdimensionen und Block-Thread-Dimensionierung bestimmt. Analog passiert dies für die Vektoraddition und Vektorskalierung, dies ist in Abb. 1 dargestellt.

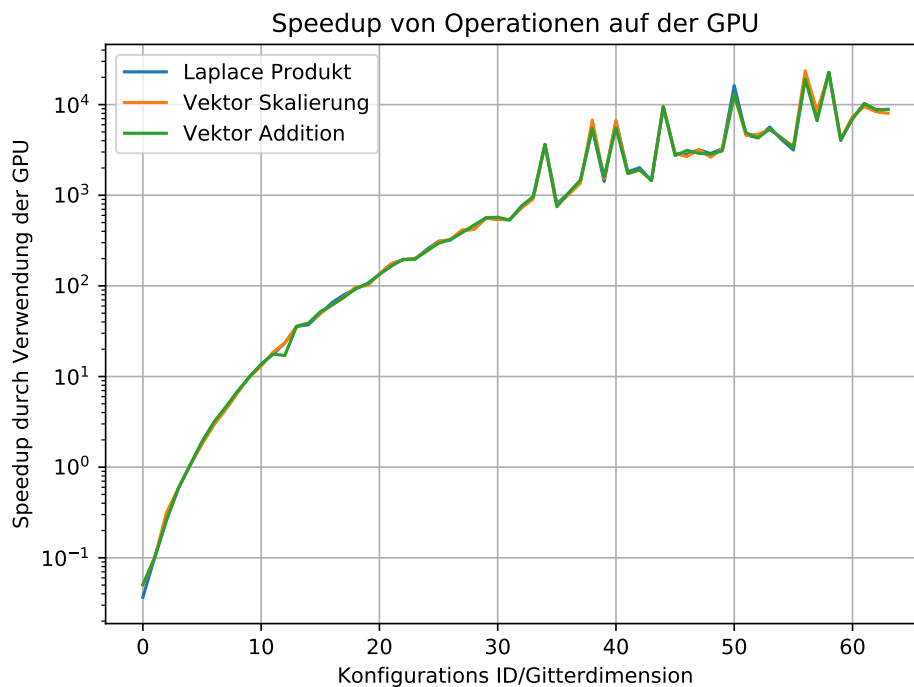


Abbildung 1: Speedup der verschiedenen Funktionen in einem Semilogplot.

Während für kleine Gitter die Funktionen auf der CPU noch schneller ausgeführt werden, ist mit steigender Gitterdimension ein Speedup in der Größenordnung von  $10^4$  erreichbar. Während in Aufgabe 1 die execution configurations mit großen Blöcken und kleinen Threadanzahlen die schnellsten Laufzeiten erreichen, sieht man an Abb. 1, dass der Speedup für große Gitterdimensionen um eine Größenordnung fluktuiert. Beispielsweise ist bei einer Gittergröße von  $60 \cdot 60$ , einer Blockverteilung von  $2 \cdot 2$  und einer Threadverteilung  $31 \cdot 31$  der Speedup bei ungefähr 20 000, während bei einer Blockverteilung von  $5 \cdot 5$

und einer Threadverteilung  $13 \cdot 13$ , für ein  $63 \cdot 63$  Gitter, der Speedup nur bei etwa 7000 liegt. In Aufgabe 1 jedoch waren gerade Kkonfigurationen mit vielen Blöcken und dafür wenigen Threads pro Block am schnellsten.