GPU Programmierung: Letztes Assignment Dokumentation
Radix Sort

Group 6
Abdelrahman Abdelhalim

Was ist das Problem? Was ist der grundlegende Algorithmus?
Radix-Sortierung ist ein Sortieralgorithmus mit Nullvergleichen, der die Elemente sortiert, indem er zunächst die Ziffern desselben Exponenten (10 oder einen anderen gewählten Wert) zählt. Anschließend werden die Elemente nach der zuvor ermittelten Anzahl sortiert.

Wie habt ihr den Algorithmus parallelisiert?
Der Teil, der sich leicht parallelisieren ließ, war der Zählalgorithmus. Die parallele Methode führt also einen Thread für jedes Element im Zählarray aus. Jeder Thread geht einmal durch das Array, das sortiert werden muss und zählt, wie viele Elemente mit dem entsprechenden Rest, wenn durch den Exponenten geteilt.

Was für Probleme gab es bei der Implementierung?
- Das erste Problem, mit dem ich konfrontiert wurde, war, dass ich 10 als Exponent wählte, so dass nur 10 Threads liefen, was dazu führte, dass die parallelisierte Implementierung zweimal langsamer war als die serielle.

- Zunächst war der Exponent ein int, was zu dem Problem führte, dass nach der Multiplikation die Grenze des int überschritten wurde.
(int): 1.000.000.000 * 10 = 1.410.065.408

- Ein Problem war auch, dass das Array von der CPU zur GPU und umgekehrt kopiert werden musste, was viel Zeit in Anspruch nimmt und die Methode verlangsamt.

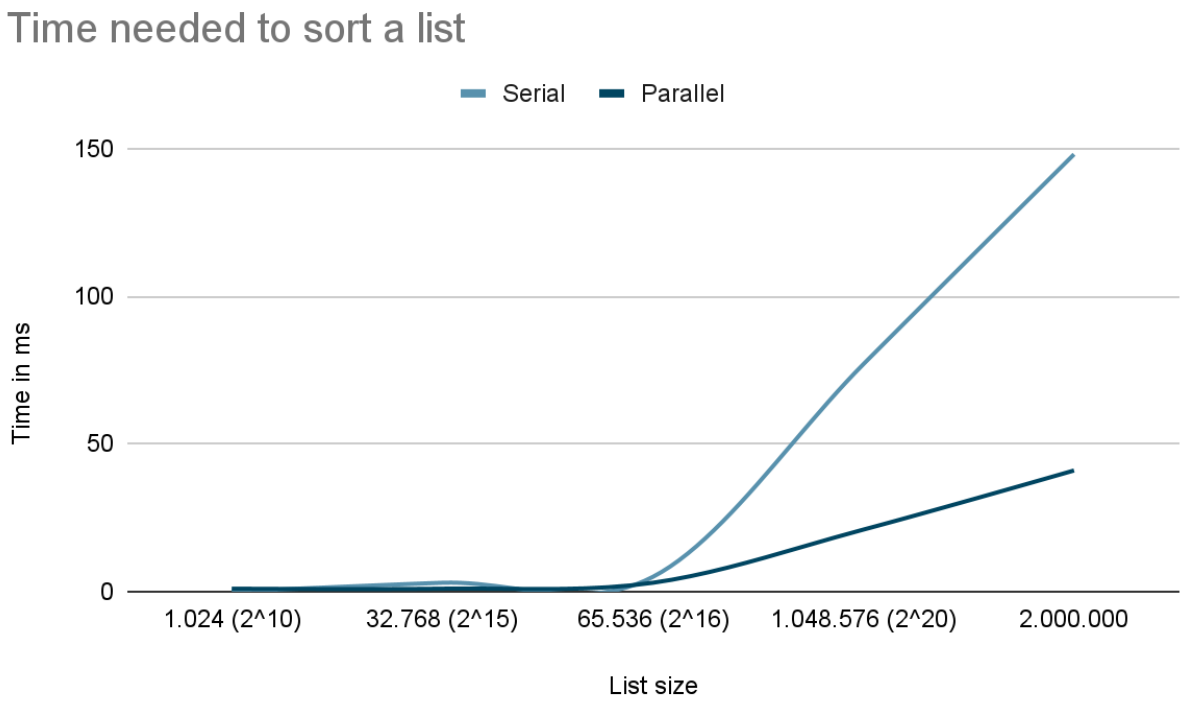
- Nach der Wahl von 256 als Exponent, was bedeutet, dass der Zählalgorithmus jedes Byte in der binären Darstellung der Zahlen im Array bearbeitet, war die Zeit, die für die Sortierung der Liste benötigt wurde, genauso lang wie die Zeit, die die serielle Methode benötigte.

Welche Optimierungen habt ihr vorgenommen?
- Um zu vermeiden, dass die Zahl nach der Multiplikation falsch ist, habe ich für den Exponenten den Typ long verwendet.
(long): 1.000.000.000 * 10 = 10.000.000.000

- Der Exponent, mit dem die Methode jetzt arbeitet, ist 65536 (256*256), was zwei Bytes entspricht. Und sie arbeitet jetzt fast viermal so schnell wie die serielle Methode.

- Die letzte Optimierung, die ich versucht habe zu tun, aber nicht ganz erreichen konnte, ist der Kern des Zählalgorithmus innerhalb der Radix-Sortierfunktion, so dass ich nicht das Array von der CPU auf die GPU und umgekehrt kopieren muss.

Was sind die Ergebnisse (Output, Zeitmessungen, Speedup etc.) eurer Implementation?
Ein Vergleich zwischen der Zeit, die die serielle Methode und die parallele Methode benötigen, ohne die Kopierzeiten.



List size	Serial	Parallel
1.024 (2^10)	0 ms	1 ms
32.768 (2^15)	3 ms	1 ms
65.536 (2^16)	5 sm	3 ms
1.048.576 (2^20)	77 ms	21 ms
2.000.000	148 ms	41 ms