TECHNISCHE UNIVERSITÄT DRESDEN

ZENTRUM FÜR INFORMATIONSDIENSTE
UND HOCHLEISTUNGSRECHNEN
PROF. DR. WOLFGANG E. NAGEL

Komplexpraktikum "Paralleles Rechnen"
A - Stringmanipulationen mit Intrinsics

Bengt Lennicke

Dresden, 4. Dezember 2023

Inhaltsverzeichnis

1 Aufgabenstellung

Implementieren Sie eine sequentielle und eine SIMD-parallele (mittels Intrinsics für einen Prozessor, der AVX2, AVX und FMA unterstützt) Variante für folgende String-Funktionen:

```
/* turns string "string" (with length len_string) to uppercase */
/* returns 1 if there has been an error, 0 if there has been no error */
int toUppercase(char* string, int len_string)

/* turns string "string" (with length len_string) to lowercase */
/* returns 1 if there has been an error, 0 if there has been no error */
int toLowercase(char* string, int len_string)

/* counts the appearences of character "c" in string "string" */
/* (with length len_string) */
/* returns -1 if there has been an error, and the number of appearences*/
/* if there has been no error */
int countChar(char* string, int len_string, char c)
```

- Beschreiben Sie für diese Funktionen die asymptotische Zeitkomplexität.
- Messen und Vergleichen Sie die Ausführungszeiten für sequentielle und SIMD-parallele Ausführung für Strings der Länge 10.000, 100.000, 1.000.000 und 100.000.000 .
- Nutzen Sie dafür die "romeo"Partition von taurus.
- Führen Sie jeweils 20 Messungen durch und analysieren Sie die Ergbenisse mit geeigneten statistischen Mitteln.

2 Auswertung

2.1 Zeitkomplexität

Die sequentiellen Funktionen für 'uppercasing', 'lowercasing' von Strings und dem Zählen von bestimmte Buchstaben in einem String sind in der Datei ßtring_manipulation_seq.c".
Die Funktionen heißen "toUppercaseSeq", "toLowercaseSeqünd "countCharSeq". Ich bin hier von den Namen der Aufgabenstellung abgewichen, damit ich den sequentiellen und paralleln Ansatz in einer Main Datei gleichzeitig importieren/nutzen kann.
Alle drei Funktionen arbeiten mit einem while loop, in welchem jedes Zeichen bearbeitet wird und anschließend der Pointer auf das nächste Zeichen bewegt wird. Damit hängt die Bearbeitungszeit linear von der Stringlänge ab. Die asymptotische Zeitkomplexität ergibt sich damit zu: O(n).

Die parallelen Funktionen sind in der Datei ßtring_manipulation_par.cünd heißen "toUppercasePar", "toLowercaseParünd "countCharPar". Die Funktionen laden jeweils 32 Zeichen des Eingabe Strings in ein 256-bit Register. Wobei der nicht-durch-32-teilbare Rest in einen neu allokierten String geladen wird, welcher dann in das Register geladen wird. Die Funktionen zum 'uppercasing' etc. werden dann auf die jeweilingen Register angewendet. Damit hängt die Bearbeitungszeit davon ab wie viele 32-Charakter Blöcke existieren bzw. damit von der Länge des Strings. Die asymptotische Zeitkomplexität ist also ebenfalls: O(n).

Das beschriebene Zeitverhalten ist auch in den folgenden drei Grafiken zu erkennen. Hier wird die durchschnittlich benötigte Rechenzeit in Abhängigkeit von der Stringlänge gezeigt. Die logarithmischen Skalen wurden gewählt, weil sonst die Messpunkte von 10k, 100k und einer Million sehr dicht zusammen liegen im Vergleich zu 100 Millionen.

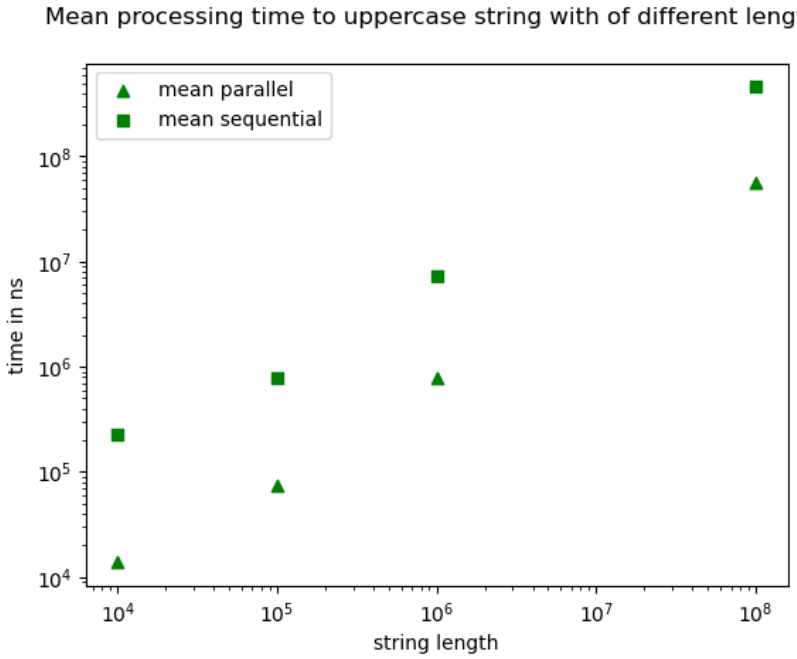

Abbildung 1: Durchschnittliche Durchführungszeit von toUppercase() in Abhängigkeit von der Stringlänge.

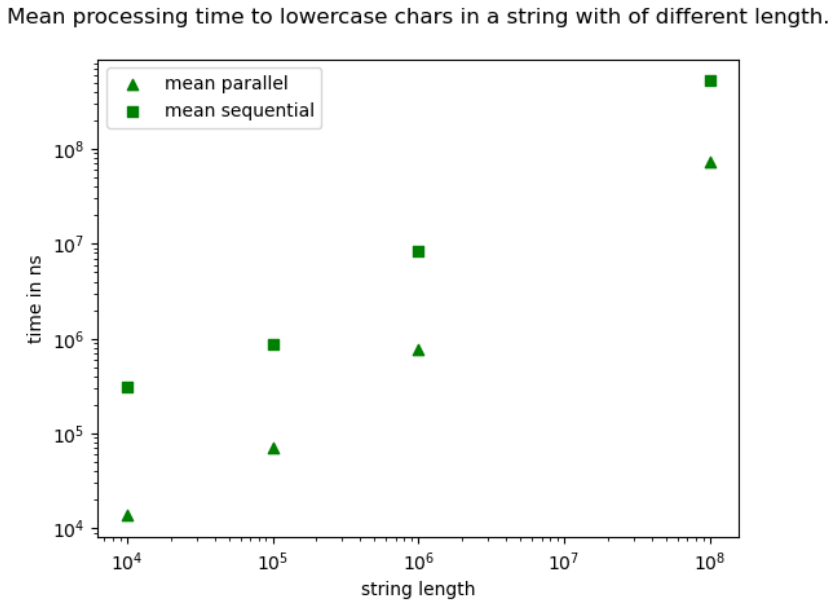

Abbildung 2: Durchschnittliche Durchführungszeit von toLowercase() in Abhängigkeit von der Stringlänge.

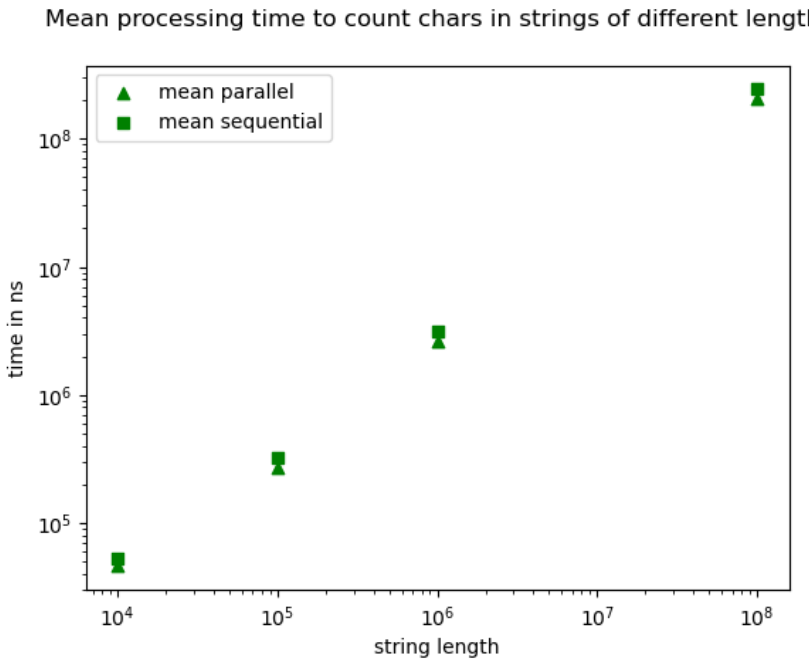

Abbildung 3: Durchschnittliche Durchführungszeit von countChar() in Abhängigkeit von der Stringlänge.

2.2 Ausführungszeiten

2.2.1 toUppercase

In der Tabelle 2.2.1 sind die Zeitmessungen für die toUppercase() Funktionen zusammengefasst. Es ist deutlich zu sehen, dass die sequentielle Ausführung in für jede Stringlänge mehr Zeit benötigt als die Umsetzung mit SIMD.

String Länge	parallel in ns		sequentiell in ns	
	Mittelwert	Standardabweichung	Mittelwert	Standardabweichung
10000	13812.89	2438.83	229699.60	75923.16
100000	74628.35	37815.71	791077.37	701784.15
1000000	781054.95	301556.31	7308699.36	3514745.39
100000000	56888725.52	5992509.42	458731319.02	61956346.45

2.2.2 toLowercase

In der Tabelle 2.2.2 sind die Zeitmessungen für die toLowercase() Funktionen zusammengefasst. Es ist deutlich zu sehen, dass die sequentielle Ausführung in für jede Stringlänge mehr Zeit benötigt als die Umsetzung mit SIMD.

String Länge	parallel in ns		sequentiell in ns	
	Mittelwert	Standardabweichung	Mittelwert	Standardabweichung
10000	13718.13	2732.00	311293.96	435716.85
100000	70663.97	33116.91	868215.45	790953.91
1000000	776718.16	328559.81	8446686.96	4419921.43
100000000	73948537.61	9256554.24	521830060.76	58353286.64

2.2.3 countChar

In der Tabelle 2.2.3 sind die Zeitmessungen für die countChar() Funktionen zusammengefasst. Es ist zu sehen, dass die sequentielle Ausführung in für jede Stringlänge mehr Zeit benötigt als die Umsetzung mit SIMD. Hier ist der Unterschied allerdings nicht so deutlich wie bei den vorherigen Funktionen. Hier würde es sich lohnen nach einer effizienteren Lösung zu suchen als dem aktuellen Ansatz.

String Länge	parallel in ns		sequentiell in ns	
	Mittelwert	Standardabweichung	Mittelwert	Standardabweichung
10000	46062.37	5738.73	53350.61	6362.82
100000	269818.84	106161.31	322522.16	122342.74
1000000	2617943.71	559510.81	3132603.61	554707.70
100000000	206280004.77	15598865.64	244304383.10	12724125.35

2.3 Vergleich