Universität Trier
Systemsoftware

Elias Kronwitter
1352465

Übung 1

Aufgabe 1

Ein Programm ist eine in einer Programmiersprache geschriebene, ausführbare Datei. Ein Programm wird zu einem Prozess, wenn es vom Betriebssystem gestartet wird. Ein Prozess ist eine Instanz eines Programms, die gerade ausgeführt wird. Wenn ein Programm gestartet wird, erstellt das Betriebssystem einen Prozess und stellt ihm Ressourcen zur Verfügung. Ein Thread ist die kleinste Ausführungseinheit innerhalb eines Prozesses. Ein Prozess kann einen oder mehrere Threads enthalten, die parallel oder quasi-parallel arbeiten. Threads teilen sich den Speicher und die Ressourcen des Prozesses und können somit effizient miteinander kommunizieren.

Aufgabe 2

Parallelisierbarer Anteil → Anzahl Kerne ↓	25%	50%	75%
1	0,75 + (0,25/1) = 1	0,5 + (0,5/1) = 1	0,25 + (0,75/1) = 1
2	0,75 + (0,25/2) = 0,875	0,5 + (0,5/2) = 0,75	0,25 + (0,75/2) = 0,625
4	0,75 + (0,25/4) = 0,8125	0,5 + (0,5/4) = 0,625	0,25 + (0,75/4) = 0,4375
8	0,75 + (0,25/8) = 0,78125	0,5 + (0,5/8) = 0,5625	0,25 + (0,75/8) = 0,34375

Aufgabe 3

Name des Branches: s2elkron

Nicht parallelisiertes Programm:

```
time python SySo/python/src/count_lines_in_files.py ~/Documents ".*\.txt$"
Total number of lines: 222618
python SySo/python/src/count_lines_in_files.py ~/Documents ".*\.txt$"  0,36s user 2,40s system 38% cpu 7,268 total
```

Parallelisierte Variante – 1 Thread:

```
time python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$" --threads 1
Total number of lines: 222618
python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$  0,37s user 2,54s system 39% cpu 7,423
total
```

Parallelisierte Variante – 2 Threads:

```
time python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$" --threads 2
Total number of lines: 222618
```

Elias Kronwitter
1352465

python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$ 0,41s user 2,75s system 45% cpu 6,939 total

Parallelisierte Variante – 4 Threads:

time python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$" --threads 4
Total number of lines: 222618
python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$ 0,40s user 2,23s system 40% cpu 6,531 total

Parallelisierte Variante – 8 Threads:

time python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$" --threads 8
Total number of lines: 222618
python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$ 0,31s user 2,33s system 41% cpu 6,352 total

Parallelisierte Variante – 12 Threads:

time python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$" --threads 12
Total number of lines: 222618
python SySo/python/src/count_lines_in_files_parallel.py ~/Documents ".*\.txt$ 0,35s user 2,06s system 38% cpu 6,297 total

Diese Zeiten sind die Mediane der Ergebnisse. Beim Testen gab es große Schwankungen. Ich vermute, dass es womöglich daran liegt, dass der Ordner auf einer Festplatte ist und somit die Zugriffszeiten eine große Rolle spielen. Insgesamt scheinen die Ergebnisse einer logarithmischen Kurve zu folgen, wie bei Amdahls Gesetz zu erwarten ist. Ein Grund warum bei einem Thread das parallelisierte Programm langsamer als das ursprüngliche sequenzielle Programm ist, ist womöglich der zusätzliche Overhead.