lea.schoenberger [☺] tu-dortmund.de
kuan-hsun.chen [☺] tu-dortmund.de
nils.hoelscher [☺] tu-dortmund.de

Übung zur Vorlesung
Rechnerarchitektur
Sommersemester 2019

Übungsblatt 2

Bearbeitung ab Mittwoch, 24. April 2019

2.1 Theorie

Ein sequentielles Programm $\mathcal{P}$ lässt sich in 5 Bereiche A bis E unterteilen, die aufgrund von Abhängigkeiten in dieser Reihenfolge ausgeführt werden müssen. Der Anteil der Bereiche an der Laufzeit ist in Tabelle 1 angegeben. Die Bereiche A, C und E lassen sich nicht parallelisieren. Bereich B kann in maximal 4 parallel ausführbare Einheiten transformiert werden. Für die Parallelisierbarkeit von Bereich D gibt es keine Beschränkung.

a) Wie viele Prozessorkerne werden benötigt, um einen Speed-Up von mindestens 4 zu erhalten? Geben Sie an, welche Gesetzmäßigkeit Sie angewendet haben.

b) Ausgehend von der Lösung in Teil a) soll in Bereich D nun ein doppelt so großes Problem gelöst werden. Die Gesamtlaufzeit des parallelisierten Programms soll sich hierbei nicht ändern. Benutzen Sie Gustavsons Gesetz, um den resultierenden Speed-Up *bezüglich der Gesamtlaufzeit des parallelisierten Programmes aus Teil a)* zu berechnen.

Bereich	A	B	C	D	E
Laufzeitanteil	2%	20%	5%	70%	3%

Tabelle 1: Anteile der Bereiche an der Programmlaufzeit

2.2 OpenMP – erste Schritte

Betrachten Sie folgendes C-Programm:

```
int main()
{
#pragma omp parallel
{
printf("Hallo Welt\n");
}
return 0;
}
```

a. Übersetzen Sie das Programm und führen Sie es aus. Vergessen Sie nicht, “-fopenmp” zu verwenden. Wie oft wird “Hallo Welt” ausgegeben?

b. Ändern Sie den Code so, dass das Programm die Nummer des jeweiligen Threads (ID) und die Anzahl aller Threads ausgibt. Die Ausgabe sollte wie die folgende sein:

```
Hallo Welt: Ich bin Thread Nummer x von y Threads
```

Dabei ist x die ID des Threads und y die Gesamtzahl der Threads. Ist es möglich, die Anzahl der Threads in dem Programm zu steuern?

2.3 Schleife Parallelismus

Im Repository befindet sich ein sequentielles Programm (0202.c).

a. Fügen Sie den erforderlichen Code hinzu, um die for-Schleife mit OpenMP zu parallelisieren. Messen Sie die Laufzeit des sequentiellen und parallelen Programms. Was fällt Ihnen auf?

b. Parallelisieren Sie die Schleife nur mit “#pragma omp parallel” (ohne “#pragma omp parallel for” oder "#pragma omp for“)