Paralleles Rechnen SS12

Matthias Uschok, Tim Lüdtke

Aufgabenblatt 1

Aufgabe 1

1. Compileroptimierungen: Führen dazu, dass der gleiche semantische Ablauf mit weniger CPU-Befehlen und optimalerweise mit weniger Rechenoperationen auskommt.

2. Architektur der Hostmaschine: Je mehr CPUs und/oder Kerne, desto mehr Threads können theoretisch parallel laufen. Je breiter die Addressleitung, desto größer die Variablen, die in einer Operation geholt werden können.

3. Aktuelle Umgebung/Situation: Laufen noch andere Tasks auf der Hostmaschine, steht weniger Rechenleistung für unseren Task zur Verfügung. Außerdem entstehen Kosten durch Kontextwechsel, wenn unterschiedliche Tasks parallel laufen (und nicht seriell).

4. Betriebssystem: Abhängig vom Scheduler werden unserem Task unterschiedlich häufig und unterschiedlich bevorzugt Ressourcen zugeteilt.

Aufgabe 2

a) **Für ein komplettes Programm** bietet sich der UNIX-Befehl '*time*' an, welcher aber leider maximal Millisekunden-genau ist.

b) **Für ein kurzes Codefragment** (aber auch nahezu ein komplettes Programm) wird direkt vor und direkt nach Ausführung des Fragments jeweils die Zeit einer hochauflösenden Uhr gespeichert und die Differenz gebildet. Auf UNIX-Systemen (wie dem Rechencluster) ist nach erster Recherche die POSIX-Funktion clock_gettime() das Mittel der Wahl. Die Funktion clock_gettime() verwendet nach Möglichkeit den HPET des Systems, wenn auf dem System kein HPET vorhanden ist, wird der nächst-präzise versucht zu benutzen.

Die Überprüfung der Genauigkeit wird vorgenommen, indem ein Codefragment mit null Operationen mehrere Male gemessen wird. Das Maximum wird als Genauigkeit interpretiert. Die Genauigkeit unserer Messmethode liegt bei Ausführung durch das Batchsystem bei etwa 520 ns.

Aufgabe 3

a) + b) siehe Code

c) Es wurden jeweils fünf Messungen für vier unterschiedliche Werte von n vorgenommen. Ergebnisse sind in ex3_c.out zu finden.

d) Die gewählte Schrittweite beträgt 1000, die Ergebnisse sind in ex3_d.out zu finden. In

ex3_d.png sind die Laufzeiten über n aufgetragen und in ex3_d_flops.png sind die erreichten GFlops über n aufgetragen. Die maximal gemessene Leistung liegt bei etwa 170 GFlops.

Aufgabe 4

Es sind Unterschiede zwischen dem GNU / Intel Compiler feststellbar, die für höhere n (Anzahl der Iterationen) zunehmen. Interessanterweise ist der vom GCC erzeugte Code bei hohen n etwa ein Achtel schneller als der vom ICC erzeugte Code. Dass es Unterschiede gibt, liegt daran, dass die beiden Compiler
1. ohnehin schon unterschiedlichen Assemblercode erzeugen und
2. unterschiedlich gut optimieren.
Wir haben die Binaries für alle Aufgaben ohne Optimierungen erzeugt, was vielleicht das Ergebnis erklärt.

Die Grafik findet sich in ex4.png