Dr. Andreas Borchert
Prof. Dr. Stefan Funken
Prof. Dr. Karsten Urban
Markus Bantle, Kristina Steih

Institut für Numerische Mathematik
Universität Ulm
Sommersemester 2013

High Performance Computing – Blatt 1

(Präsenzübung 22. April 2013)

Diskussion

- *C++ Grundlagen:*
  - Was ist ein Zeiger (bzw. *pointer*)? Was ist eine Referenz?
  - Was sind Funktionsobjekte?
- *Vorbereitung Aufgabe 1:*
  - Wie wird in C++11 ein Thread angelegt?
  - Was für eine Funktion führt dieser aus?
  - Wie kann man dieser Funktion Parameter übergeben?
- *Vorbereitung Aufgabe 2:*
  - Was versteckt sich hinter `double (*f)(double)`?

    [Siehe z.B. Skript Folie 26, Stichwort *Funktionszeiger*]
  - Was ist der Zusammenhang zwischen `mt_simpson`, `SimpsonThread`, `simpson`? Was wird wann aufgerufen/instanziiert?
  - Erklären Sie Folie 29.
- *Vorbereitung Aufgabe 3:*
  - Was ist BLAS (*Basic Linear Algebra Subprograms*)?
  - Welche Parameter erhält z.B. die BLAS-Funktion `copy`? Was ist mit *"stride"* gemeint?

    [Auf der Homepage finden Sie ein Kapitel der Dissertation von Dr. Michael Lehn, der ein C++-Interface namens FLENS für u.a. BLAS-Funktionen geschrieben hat. Die Sektionen 3.1.0–3.1.1, 3.1.3–3.1.4, 3.3.2–3.3.3 helfen hier weiter.]

Aufgabe 1: Ganz simple Threads

a) Schreiben Sie ein Programm, welches n Threads erzeugt. Diese sollen jeweils eine ID-Nummer haben und diese bei Aufruf auf die Standardausgabe ausgeben.

b) Erweitern Sie Ihr Programm aus (a): Eine im Hauptprogramm angelegte Variable `int count` soll von jedem Thread jeweils um 1 erhöht werden, anschließend soll die Thread-ID und der aktuelle Wert von `count` ausgegeben werden. Geben Sie am Schluß des Hauptprogramms noch einmal den Wert von `count` aus. Was kann hier schiefgehen?

Aufgabe 2: Jetzt wird auch was gerechnet

Analog zum parallelisierten Simpson-Verfahren aus der Vorlesung soll nun eine Funktion

```
double sum(double* a, int n)
```

parallelisiert werden, welche die n Einträge eines Arrays a aufaddiert.

a) Auf der Homepage finden Sie die Datei *sum.cpp*, welche im Wesentlichen das Hauptprogramm enthält. Vervollständigen Sie die fehlenden Teile.

b) Führen Sie das Programm für verschiedene Arraylängen und verschiedene Threadzahlen aus. Was beobachten Sie?

*c) Templaten Sie die Funktion `sum` auf den Datentyp.

Aufgabe 3: Paralleles BLAS

Auf der Homepage finden Sie Implementierungen zu verschiedenen BLAS-Funktionen (*blas.h*, *blas.tcc*). Diese sollen nun parallelisiert werden.

a) Schauen Sie sich das Material an: Zusätzlich zu den BLAS-Dateien gibt es noch

- *hpc_blas_test.h*/*hpc_blas_test.tcc*: Testklasse, in der verschiedene Vektoren/Matrizen angelegt und zugehörige Tests definiert werden. Dabei werden die jeweiligen BLAS-Funktionen aufgerufen, das Ergebnis überprüft und die Laufzeiten gemessen.
- *test_mt_blas.cpp*: Das Hauptprogramm, in dem eine Testklasse angelegt und die Tests aufgerufen werden.
- *timer.h*: Eine Klasse, die einen Zeitmesser zur Laufzeitmessung zur Verfügung stellt.
- *mt_blas.h*: Hier drin stehen schon die Deklarationen für die parallelen BLAS-Funktionen.

b) Implementieren Sie die parallelen Versionen der BLAS-Funktionen aus *mt_blas.h* (analog zu *blas.tcc* sollten diese in einer Datei *mt_blas.tcc* gespeichert werden).

c) Lassen Sie die Tests laufen. Was beobachten Sie?