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 2

(Präsenzübung 29. April 2013)

Diskussion

- *C++ Grundlagen: Templates*

  Betrachten Sie die folgende Funktion:

```
template <typename T>
T sum(T* a, int n){
    T val = 0;
    for(int i = 0; i < n; ++i) val += a[i];
    return val;
}
```

  - Was steckt hinter `T`?
  - Wie könnte die Funktion `sum` im Hauptprogramm zum Beispiel aufgerufen werden?
  - Warum verwendet man Templates?

- *Vorbereitung Aufgaben 1 & 2:*
  - Was macht die Anweisung `#pragma omp parallel for`? Worauf muss man bei der Verwendung achten?
  - Was macht die Anweisung `#pragma omp parallel`? Was für Probleme können dabei auftreten?
  - Was für Parallelisierungsklauseln gibt es zum Beispiel?
  - Wie findet man heraus, wie viele Threads erzeugt wurden? Wie kann man bestimmte Aufgaben nur im Master-Thread ausführen (also dem Thread des Hauptprogramms)?

- *Vorbereitung Aufgabe 2:*
  - Was machen die folgenden BLAS-Funktionen: `axpy`, `dot`, `gecopy`, `gemv`?

Aufgabe 1: Wieder eine parallele Funktion `sum`

Analog zum letzten Blatt soll wieder die Funktion

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

parallelisiert werden, diesmal aber mittels OpenMP.

a) Kopieren Sie die Datei *sum.cpp* von Blatt 1 und implementieren Sie diesmal die Funktion

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

durch eine mit OpenMP parallelisierte `for`-Schleife.

b) Implementieren Sie eine weitere Funktion `mt_sum_explicit`, in der Sie analog zu *omp-simpson-explicit.cpp* auf Folie 71 die Parallelisierung *explizit* durchführen.

c) Vergleichen Sie die Laufzeiten der seriellen und parallelen Implementierungen.

Hinweis: Denken Sie daran,

- den Header `<omp.h>` einzubinden,
- beim Kompilieren die zusätzliche Option `-fopenmp` anzugeben.

Aufgabe 2: Paralleles BLAS

Auch die BLAS-Funktionen sollen nun mit OpenMP parallelisiert werden. Auf der Homepage finden Sie im Wesentlichen die gleichen Dateien wie auf Blatt 1.

- Die mit Threads parallelisierten BLAS-Funktionen liegen jetzt im Namespace `mt_threads`.
- Es gibt eine zusätzliche Datei *omp_blas.h*, in der wie in *mt_blas.h* parallele BLAS-Funktionen deklariert sind, hier allerdings im Namensraum `mt_omp`.

a) Parallelisieren Sie die BLAS-Funktionen aus *omp_blas.h* mit OpenMP (analog zu *mt_blas.tcc* sollten diese in einer Datei *omp_blas.tcc* gespeichert werden).

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