Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Prof. T. Ludwig, J. Kunkel, N. Hübbe

Übungsblatt 3 zum Praktikum
Paralleles Programmieren
im SoSe 2013

Auf diesem Aufgabenzettel finden Sie wieder einige Aufgaben. Sie müssen nicht alle bearbeiten, investieren Sie aber zwischen 5-10 Stunden um sich mit einigen davon zu beschäftigen. Vermutlich genügt es von jeder Sektion eine Aufgabe zu bearbeiten.

1 OpenMP

Der OpenMP hat sich als Standard zur Programmierung von gemeinsamen Speicher weit verbreitet (Shared-Memory Programmierung, Global-Address-Space GAS).

1.1 OpenMP Hello-World

Schreiben Sie ein Programm, welches eine Anzahl an Threads startet und für jeden Thread eine Zeile mit `hello World von Thread NUMMER` ausgibt. Die Thread-Anzahl sollte hierbei als Übergabeparameter angenommen werden.

1.2 Aggregation

Schreiben Sie ein Programm, welches von der Kommandozeile beliebig viele Parameter im Zahlenformat annimmt. Diese Zahlen sollen fair auf die vorhanden Threads aufgeteilt werden (kein Threads muss 2 Zahlen mehr berechnen als ein anderer, maximal 1 Zahl). Diese Threads addieren dann ihre zugeteilten Zahlen in einer Schleife. Der Master-Thread gibt die Zahl am Ende aus.

1.2.1 Addition mittels seriellen Bereichen

Verwenden Sie einen kritischen Abschnitt zur Addition (`critical`)

1.2.2 Verwendung von Reduktionsvariablen

Verwenden Sie eine Reduktionsvariable (`reduction`) anstelle eines seriellen Bereiches.

1.2.3 Verwendung von Atomaren Auswertungen

Verwenden Sie eine Atomare Auswertung (`atomic`) zur finalen Addition der Zahlen.

2 Julia Mengen

Verwenden Sie OpenMP zur Parallelisierung der Berechnung der Julia Mengen (siehe Aufgabenblatt 2).

3 Matrix-Matrix Multiplikation

Verwenden Sie OpenMP zur Parallelisierung der Matrix-Matrix Multiplikation (siehe Aufgabenblatt 1 und 2).

4 Leistungsvergleiche

Schreiben Sie ein kleines Benchmark-Programm um etwas mehr über die Leistungsfähigkeit von OpenMP zu erfahren. Stoppen Sie hierfür die Zeit die eine Anzahl von Operationen benötigen und geben Sie die Operationen pro Sekunde aus.
Schachteln Sie zwei Schleifen ineinander und lassen Sie die innere Schleife mittels `parallel for` Direktive parallelisieren. Ein kleiner Rumpf z.b. Addition einer Zahl wird auch benötigt, damit der Compiler den Code nicht weg optimiert.
Bestimmen Sie mit ähnlichem Aufbau den Zusatzaufwand einmal für Atomare, für serielle Bereiche und für Reduktionen.
Starten Sie das Programm mehrfach mit 1-4 Threads, notieren Sie die Zeiten und erreichte Operationen pro Sekunde. Um Aussagekräftige Ergebnisse zu erreichen sollte das Programm nicht kürzer als 10 Sekunden laufen.
Wenn Sie zusätzlich den Aufwand für die einzelnen Operationen unter OpenMP berechnen wollen, so lassen Sie die selben beiden Schleifen ohne Direktiven laufen und messen Sie die Zeit, die dann benötigt wird. Vergleichen Sie die erzielte Zeit des sequentiellen Codes mit der Zeit des parallelen Codes für 1, 2 und 4 Threads und erstellen Sie Speedup Diagramme.

5 Leistungsvergleiche von MPI mit OpenMP

Falls Sie eines der Programme (z.B. Julia Mengen oder Matrix-Matrix Multiplikation) sowohl mit OpenMP als auch mit MPI geschrieben haben, so vergleichen wir die Leistungsfähigkeit. Hierfür starten Sie die Programme auf nur einem Knoten mit 1-4 Prozessen bzw. Threads. Notieren Sie die erzielten Laufzeiten und Prozesszahlen in Sekunden für ein Programm ihrer Wahl, mit 3 Nachkommastellen in einer Textdatei. Das Programm sollte sinnvolle Zeiten ausgeben, d.h. für einen Prozess sollte die Berechnung schon eigene Minuten dauern. Erstellen Sie ein Speedup-Diagramm für 1-4 Threads bzw. Prozesse mit den Beispieleingaben.