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

Übungsblatt 2 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 MPI-I/O

MPI-2 definiert eine Schnittstelle zur Ein/Ausgabe von Dateien. Diese erlaubt portabel Dateien auszutauschen und den parallelen Zugriff auf die Dateien.

1.1 Matrix-Matrix-Multiplikation

Erweitern Sie ihr Matrix-Matrix Programm, so dass dieses MPI-I/O zur Eingabe bzw. Ausgabe der Daten nutzt. Erzeugen Sie eine Dateisicht um alle Daten der Matrix auf einmal einlesen zu können.
Sollten im Beispiel kollektive- oder individuelle MPI-Aufrufe verwendet werden?

2 MPI-PGAS

Partitioned Global Adress Space (PGAS), ist ein Model, bei dem jeder Prozess explizit auf Speicher aller anderen Prozesse zugreifen kann. Im Modell mit Nachtrichtenaustausch musste sowohl Sender als auch Empfänger teilnehmen. Bei PGAS kann ein Prozess Daten eines zweiten Prozesses lesen oder schreiben, ohne dass der zweite Prozess Kentniss von der Operation bekommt.
Schauen Sie folgende Operationen an: `mpi_put()`, `mpi_get()`, `mpi_win_create()`, `mpi_win_free()`, `mpi_win_fence()`, `MPI_Accumulate ()`.

2.1 Matrix-Matrix Multiplikation

Programmieren Sie die Matrix-Matrix Multiplikation mit einseitiger Kommunikation unter MPI-2.

2.2 Julia Mengen

Betrachten wir eine Iterationsvorschrift der Form $x_{n+1} = x_n + c$ im Raum der komplexen Zahlen. Wobei x_0 als Initalwert gegeben ist. Ebenfalls sei c gegeben.
Um Julia Mengen für ein gegebenes $c \in \mathbb{C}$ zu visualisieren, kann man auf der X-Achse den Realanteil und auf der Y-Achse den Imaginärteil der Startwerte (x_0) z.B. zwischen -0.5 und 0.5 nutzen. Die Farbwerte an einer Position geben an, wieviele Iterationen gebraucht werden, bis die Iterationsvorschrift divergiert (gegen ∞ geht). Es gibt daneben auch Startwerte, bei denen die Iterationsvorschrift unendlich oft angewendet werden, d.h. es divergiert niemals. Das Program wird daher nach einer Anzahl an Iterationen (maximale Iterationsanzahl) abbrechen und die maximale Iterationszahl für die Position setzen. Um festzustellen dass die Iteration divergiert kann man den Absolutbetrag verwenden. Sobald $|x_n| > 4.0$ geht, so kann man abbrechen, denn die Vorschrift wird divergieren. Die Anzahl der Iterationen werden dann in eine Matrix (entsprechend der Position) gespeichert.
Sequentiellen Beispielcode (in C) für die Julia-Mengen finden Sie auf unserer Webseite. Das Programm erstellt Grafiken (Bitmaps) entsprechend der Parameter.
Schauen Sie z.B. die Grafik an für: `./juliaMengen -0.39 0.6 test.bmp 0 0 0 1000` und `./juliaMengen -0.39 0.6 test.bmp 11 0 0 1000`.

Das Programm kann einerseits mehrfarbige Bilder erzeugen, wobei für jeden Punkt bis zu 2^{24} Iterationen durchgeführt werden müssen (wenn der Absolutwert nicht divergiert). In einem einfarbigen Bild werden nur 2^8 Iterationen durchgeführt.
Im Farbbild kann es erwartungsgemäß sehr lange dauern bis alle 2^{24} Iterationen abgeschlossen sind. Es gibt aber auch Positionen die nach wenigen Iterationen divergieren.
Ein Beispiellauf bei dem nicht alle Punkte divergieren: `./juliaMengen -0.39 0.5 test.bmp 0 0 0 10`.
Beispiele für schöne Parameter finden Sie unter `http://de.wikipedia.org/wiki/Julia-Menge`.

Parallelisierung mittels Gebietszerlegung und PGAS Parallelisieren Sie das Program mittels Gebietszerlegung. Ein Masterprozess sammelt die Iterationszahlen von allen Gebieten in einer Matrix. Nutzen Sie lediglich einseitige MPI-Aufrufe um die Ergebnisse in die Matrix zu schreiben.
Rechnen alle Prozesse immer gleich lange? Wodurch entsteht die Lastungleichheit?

Parallelisierung mit Lastausgleich Parallelisieren Sie das Program mit Lastausgleich. Da es Punkte gibt die erst spät divergieren werden für jeden Punkt bis zu 2^{24} Iterationen durchgeführt. Diese Lastungleichheit wollen wir nun beseitigen.
Verteilen Sie hierzu die zu berechnenden Punkte dynamisch auf die Prozesse. Dazu können Sie einen Masterprozess die Zuteilung übernehmen lassen, dieser Prozess könnte die zu berechnenden Werte Blockweise z.b. in Blöcken von 100 Werten an die Workerknoten übergeben. Sobald ein Prozess seine zugeteilten Werte bearbeitet hat und die Ergebnisse an den Master zurück schickt.
Verwenden Sie beliebige MPI-Aufrufe. Wenn Sie mit dem PGAS Modell weiter programmieren wollen, so benötigen Sie vermutlich `MPI_Win_lock()`.

3 Speedup Diagramme

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 dieser schon eigene Minuten dauern. Erstellen Sie ein Speedup-Diagramm für 1-12 Prozesse mit den Beispieleingaben.