Otto-von-Guericke-Universität Magdeburg
Max-Planck-Institut for Dynamics of Complex Technical Systeme
Computational Methods in Systems and Control Theory

Dr. Jens Saak, Dipl.-Math. Martin Köhler
Webseite: http://www.mpi-magdeburg.mpg.de/mpcsc/lehre/2013_SS_SC/

Sommersemester 2013

Scientific Computing 2
Hausaufgabenblatt 6

Ausgabe: 26. Juni 2012 **Abgabe:** 9. Juli 2012

Hinweis: Als MPI Implementierung eignet sich am besten OpenMPI[1]. Unter Debian/Ubuntu-artigen Linuxdistributuionen kann dies mit Hilfe der Pakete `openmpi-bin` und `libopenmpi-dev` installiert werden.

Aufgabe 1: (4 Punkte)

Schreiben Sie ein "Hello World" Programm in MPI. Jede Instanz des Programms soll dabei seine Prozess-Nummer (`rank`) und die Gesamtanzahl der Prozesse ausgeben. Für 4 MPI-Prozesse kann die Ausgaben beispielsweise wie folgt aussehen:

```
Hallo ich bin Prozess 0 von 4
Hallo ich bin Prozess 3 von 4
Hallo ich bin Prozess 1 von 4
Hallo ich bin Prozess 2 von 4
```

Aufgabe 2: (4 Punkte)

Schreiben Sie ein MPI Programm, welches folgenden Ablauf realisiert. Der Prozess 0 (d.h. `rank==0`) soll vom Benutzer einen Text mit maximal 100 Zeichen einlesen. Dieser wird an den Prozess 1 weiter geschickt, welcher alle Buchstaben in Großbuchstaben umwandelt und den Text an Prozess 2 zur Ausgabe weiterleitet. Sollten mehr oder weniger als 3 Prozesse gestartet werden so soll eine Warnung ausgeben werden und das Programm sofort beendet werden.

Aufgabe 3: (8 Punkte)

Schreiben Sie ein MPI Programm, welches parallel das Skalarprodukt $x^T y$ zwischen zwei Vektoren $x, y \in \mathbb{R}^n$ berechnet. Gehen Sie dabei wie folgt vor:

a.) Die Vektoren x und y werden auf dem Prozess 0 erzeugt und mit Hilfe einer Scatter-Operation an die beteiligten Prozesse verteilt. Die Dimension n kann/muss zuvor mit einer Broadcast-Operation verteilt werden. Die Dimension der Vektoren n soll durch die Anzahl der Prozesse teilbar sein.

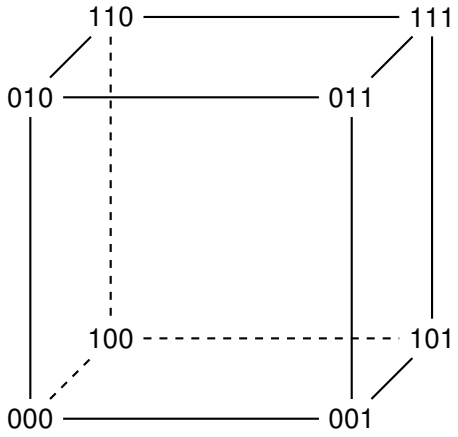
b.) Die eigentliche Rechenarbeit soll gleichmäßig unter allen Prozessen verteilt werden.

c.) Die so entstandenen Zwischenergebnisse sollen mit Hilfe einer Reduce-Operation auf Prozess 0 zusammengefasst und ausgeben werden.

Aufgabe 4: (8 Punkte)

Der Hyper-Würfel ist ein beliebtes Kommunikations-Netzwerk im wissenschaftlichen Rechnen. Wir betrachten den Würfel der Dimension 3:

[1] http://www.openmpi.org



Um laufzeiteffizient Single-Broadcast-Nachrichten an alle beteiligten Prozess zu schicken, werden die in der Vorlesung vorgestellt *Spanning-Trees* benutzt.

a.) Zeichnen Sie die Spanning-Trees für die Wurzelknoten `000` und `110`.

b.) Zeichnen Sie die beiden Spanning-Trees in einen Würfel mit Angabe in welchem Zeitschritt ein Kommunikationkanal verwendet wird. Was ist dabei festzustellen?

c.) Interpretieren Sie den Wechsel des Wurzelknotens über die $\oplus$ Operation aus der Vorlesung geometrisch.

Aufgabe 5: **(6 Punkte)**

Geben Sei folgende Funktion, welche die Gather-Operation auf einem Ring implementiert:

```
void Gather_ring (float x[], int blocksize, float y[]) {
 int i, p, my_rank, succ, pred;
 int send_offset, recv_offset;
 MPI_Status status;

 MPI_Comm_size (MPI_COMM_WORLD, &p);
 MPI_Comm_rank (MPI_COMM_WORLD, &my_rank);
 for (i=0; i<blocksize; i++)
  y[i+my_rank * blocksize] = x[i];
 succ = (my_rank+1) % p;
 pred = (my_rank-1+p) % p;
 for (i=0; i<p-1; i++) {
  send_offset = ((my_rank-i+p) % p) * blocksize;
  recv_offset = ((my_rank-i-1+p) % p) * blocksize;
  MPI_Send (y+send_offset, blocksize, MPI_FLOAT, succ, 0, MPI_COMM_WORLD);
  MPI_Recv (y+recv_offset, blocksize, MPI_FLOAT, pred, 0, MPI_COMM_WORLD,
            &status);
 }
}
```

Diese Implementierung ist nicht Deadlock frei. Suchen Sie die Ursache und geben Sie ein Deadlock Szenario an. Wie kann dieses Problem behoben werden?

Gesamtpunktzahl: 30