Bioinformatics Department
Bielefeld University

Parallele Algorithmen und Datenverarbeitung (Ü) (WiSe 2018/2019)
Marcel Friedrichs
AG Bioinformatik / Medizinische Informatik

Übungszettel 7

Aufgabe 1:

a) Nennen Sie die zentralen Verwendungszwecke eines Computerclusters.
b) Erläutern Sie die zugrunde liegende Architektur eines *Beowulf-Clusters*.
c) Wie werden die Aufgabenverteilung, die Kommunikation und der Datenaustausch auf einem *Beowulf-Cluster* realisiert?

Aufgabe 2:

a) Welche Arten von Deadlocks gibt es?
b) Wie lassen sich Deadlocks verhindern?

Aufgabe 3:

In dieser Aufgabe soll das Message-Passing-Interface (MPI) Konzept praktisch nähergebracht werden. Hierfür müssen einige Vorbereitungen getroffen werden:

1. Zur Verwendung von MPI wird eine Bibliothek benötigt. Da diese Aufgabe in Java umgesetzt wird, verwenden wir in diesem Fall „MPJ Express“ (http://mpj-express.org) welches hier in der aktuellen Version heruntergeladen werden kann: https://sourceforge.net/projects/mpjexpress/files/latest/download
2. Der Pfad der Bibliothek muss als Umgebungsvariable gesetzt werden, z.B.:
   Linux: *export MPJ_HOME="/path/to/mpj-v0_44"*
   Windows: *setx MPJ_HOME "C:\\path\\to\\mpj-v0_44"*
3. Der Java Compiler benötigt ebenfalls den Pfad zur Bibliothek für unser Programm, z.B.:
   javac -cp ".;%MPJ_HOME%/lib/mpj.jar" Program.java

MPI ermöglicht es, Aufgaben in kleinere Teile zu zerlegen und über die Prozesse zu verteilen. Hier eignet es sich oft die Prozesse in einen Master- und die restlichen in Slave-Prozesse zu unterteilen.

Beginnend mit dem Java-Code in Programm 2 schreiben Sie ein MPI Programm, welches die Summe der Zahlen von 1 bis 10.000 berechnet. Die Prozesse berechnen hierbei Teilsummen aus gleichmäßigen Stücken des Bereichs [1, 10.000] und der Master akkumuliert die Teilergebnisse zur Gesamtsumme. Die Slaves schicken die Teilergebnisse über „MPI.COMM_WORLD.Send“ an den Master, welcher diese mit „MPI.COMM_WORLD.Recv“ empfängt.

Starten Sie das Programm mit 4 Prozessen wie folgt:
%MPJ_HOME%/bin/mpjrun.bat -np 4 Program

Hinweis: Die Lösungen sollen in PDF-Form, bzw. Code bis zum Montag (10 Uhr) der jeweils folgenden Woche per Mail an mfriedrichs@techfak.uni-bielefeld.de abgegeben werden. Zu Beginn des nächsten Übungstermins werden diese in offener Runde vorgestellt und diskutiert.

```
import mpi.*;

public class Program {
  public static void main(String[] args) throws Exception {
    MPI.Init(args);
    int id = MPI.COMM_WORLD.Rank();
    int size = MPI.COMM_WORLD.Size();
    System.out.println("Process " + id + " of " + size);
    int[] sum = new int[1];

    // TODO

    If (id == 0) { // Master
      System.out.println("The sum from 1 to 10000 is: " + sum[0]);
    }
    MPI.Finalize();
  }
}
```

Programm 2