PHILIPPS-UNIVERSITÄT MARBURG D-35032 Marburg
Fachbereich Mathematik und Informatik Hans Meerwein Straße
Prof. Dr. R. Loogen
Dipl.Inf. J.Berthold

25.Mai 2005

Übungen zu „Parallele Programmierung“, SS 2005

Nr. 7, Abgabe der Aufgaben: 2.Juni 2005 vor der Vorlesung

MPI am Fachbereich 12 – Kurzfassung

Konfiguration

.tcshrc

```
setenv LAMHOME /app/lang/parallel/lam-7.0.4
setenv LAMRSH "ssh -x"
setenv PATH ${PATH}:${LAMHOME}/bin
```

Kommandos zur Steuerung des Systems in `${LAMHOME}/bin`:

- `lamboot` startet einen parallelen Rechnerverbunds.
  Beteiligte Rechner werden in einer *hostlist* angegeben.
- `lamhalt` fährt das paralleles System herunter.
- `lamnodes` gibt Informationen über das gestartete System.
- `lamclean` setzt das parallele System zurück.

Übersetzen und Starten von Programmen

- `mpicc` ist der MPI-Compiler, er versteht `gcc`-Optionen
- `mpirun` startet ein MPI-Programm (Zahl der Instanzen angegeben!)

Beispiel:

```
berthold@kitwe:VLparprog05/C> mpicc -O2 -o helloMPI HelloMPI.c
berthold@kitwe:VLparprog05/C> lamboot lamhosts

LAM 7.0.4/MPI 2 C++/ROMIO - Indiana University

berthold@kitwe:VLparprog05/C> mpirun N helloMPI
       ...(Programmausgaben)...

berthold@kitwe:VLparprog05/C> lamhalt

LAM 7.0.4/MPI 2 C++/ROMIO - Indiana University

berthold@kitwe:VLparprog05/C>
```

Bitte wenden!

Aufgaben

7.1 CREW-Lösung mit synchroner Kommunikation und `in` — 5 Punkte

Programmieren Sie eine Leser-Schreiber-Synchronisation analog zur Version aus Aufgabe 4.1, welche mit einem zentralen Zähler-Verwalter und der `in`-Anweisung mit synchroner Kommunikation arbeitet.

Die `in`-Anweisung ermöglicht die Angabe von zusätzlichen Bedingungen, welche beim Empfang einer Nachricht gelten müssen. Mit diesen Bedingungen können die bedingten Anweisungen im Code der Leser und Schreiber modelliert werden.

7.2 Odd-Even-Transposition Sort in C und MPI — 7 Punkte

Implementieren Sie das parallele Sortierverfahren *Odd-Even-Transposition Sort* aus Aufgabe 6.2 in C und MPI. Ihr Programm soll die folgenden Eigenschaften haben:

- Die Anzahl der Sortierprozesse soll der Zahl der mit `mpirun` gestarteten Instanzen des Programms entsprechen.
- Einer der Prozesse liest die zu sortierenden Elemente, hier Int-Werte, zeilenweise aus einer Datei.
- Die Elemente werden (während des Lesens oder danach?) mit Nachrichten an die anderen Instanzen verteilt.
- Alle Prozesse sortieren gemeinsam die Liste mit Hilfe von Odd-Even-Transposition Sort, wobei jeweils ganze Listen ausgetauscht werden (Lösung b aus A. 6.2).
- Nach der Sortierung werden alle Elemente wieder auf dem Knoten gesammelt, der die Datei gelesen hat.

Sie können mit einfachen MPI-Sende- und Empfangsoperationen oder mit kollektiven Operationen in MPI arbeiten.

Auf der Vorlesungsseite finden Sie ein sequenzielles Bubblesort-Programm als Vorlage.