PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen
J.Berthold

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

10.Mai 2007

4. Übung zur Vorlesung "Parallele Algorithmen", Sommer 07

Abgabe: 16.Mai 2007 vor der Vorlesung

Wegen des Feiertags (Christi Himmelfahrt, 17.5.) in der nächsten Woche ist dieses Blatt **bereits am Mittwoch, den 16.5.2007 abzugeben**.

Aufgaben

4.1 **Analyse paralleler Programme nach Messergebnissen** [6 Punkte]

Ein paralleles Programm sei in zwei unterschiedlichen Versionen implementiert. Mit verschiedenen Prozessorzahlen wurden für die beiden Programme in Tests die folgenden Laufzeiten gemessen:

Prozessoren / Laufzeiten (*sec.*)

PE	Version 1	Version 2
1	471.10	450.00
2	239.83	228.49
4	124.60	117.90
6	86.36	81.19
8	67.33	62.94
10	55.95	52.09
12	48.39	44.93
14	43.01	39.87
16	39.00	36.14
18	35.88	33.29
20	33.40	31.05

(a) Ermitteln Sie Speedup und Effizienz für die jeweiligen Messreihen und stellen Sie sie grafisch dar.

(b) Analysieren Sie die Ergebnisse mit Hilfe der Karp-Flatt-Metrik. Welche Abhängigkeit von der Prozessorzahl ist jeweils zu vermuten?

(c) Welches Programm eignet sich besser, wenn im produktiven Einsatz mehr Prozessoren verwendet werden können?
Schätzen Sie anhand Ihrer Daten aus b) das Verhalten auf 128 Prozessoren ein.

4.2 **Paralleles Sortieren** Das Verfahren *Odd-Even-Transposition Sort* ist eine parallele Variante von Bubble-Sort. Dem i. Prozessor wird das Element a_i zugewiesen. Der Algorithmus besteht aus zwei sich wiederholenden Phasen: [6 Punkte]

- Beim Ungerade/Gerade-Austausch vergleicht jeder k. Prozessor (k ungerade) sein Element mit demjenigen seines rechten Nachbarn (sofern vorhanden) und vertauscht die Elemente falls dies nötig ist.
- Beim Gerade/Ungerade-Austausch vergleicht jeder k. Prozessor (k gerade) sein Element mit dem Element seines rechten Nachbarn und vertauscht diese falls nötig.

(a) Spezifizieren Sie Odd-Even-Transposition Sort als PRAM-Algorithmus.

(b) Bestimmen Sie die parallelen Kosten des Verfahrens.
Wie viele parallele Schritte benötigt der Algorithmus? Wie kann man auf einfache Weise mit weniger Prozessoren auskommen, ohne die Grundstruktur des Algorithmus zu ändern?