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

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

26.April 2007

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

Abgabe: 10.Mai 2007 vor der Vorlesung

Aufgaben

3.1 Broadcast in verschiedenen Vernetzungstopologien — 5 Punkte

Schreiben Sie ein MPI-Programm, mit dem ein Feld von Daten gleichmäßig auf alle Prozesse verteilt wird (gehen Sie analog zur Broadcast-Operation vor). Die Daten seien anfangs in Feld `a[]` in Prozess 0 und werden in ein zweites Feld `b[]` in allen Prozessen verteilt.

	P_0	P_1	P_2	P_3	P_4	P_5
$a:$	$1,\ldots,12$					

$\Rightarrow$

	P_0	P_1	P_2	P_3	P_4	P_5
$b:$	$1,2$	$3,4$	$5,6$	$7,8$	$9,10$	$11,12$

MPI definiert eine spezielle kollektive Operation `MPI_Scatter` für diesen Vorgang. Diese soll hier aber *nicht* verwendet werden. Implementieren Sie statt dessen den Algorithmus für die Vernetzungstopologien Shuffle-Exchange und Toroid

Sie können davon ausgehen, dass die Aufteilung "aufgeht", d.h. bei n Daten und p Prozessen ist n immer durch p teilbar.

3.2 Untere Schranken für paralleles Sortieren — 3 Punkte

Begründen Sie die Korrektheit der im folgenden angegebenen unteren Schranken für das Sortieren von n Elementen auf verschiedenen Netzwerken mit jeweils n Knoten. Vor und nach dem Sortiervorgang sollen die zu sortierenden Elemente gleichmäßig verteilt sein, d.h. ein Element pro Prozessor.

(a) $\Omega(n)$ auf einem eindimensionalen Gitter

(b) $\Omega(\sqrt{n})$ auf einem zweidimensionalen Toroid

(c) $\Omega(\log n)$ auf einem Shuffle-Exchange-Netzwerk

3.3 Speedup-Schätzungen — 4 Punkte

(a) Ein paralleler Algorithmus erfordere bei Problemgröße n $14 \cdot n$ Schritte sowie eine sequenzielle Vorverarbeitung der Eingabedaten, welche $(\log_2 n)^3$ Schritte benötigt. Wie viele Prozessoren müssen jeweils (mindestens) eingesetzt werden, um für ein Problem der Größe $n = 1024$ einen Speedup von 10 zu erzielen?

(b) Nach dem Gesetz von Amdahl wird der maximal erreichbare Speedup eines parallelen Algorithmus durch den sequenziellen Anteil $f \in [0, 1]$ begrenzt, insbesondere ist der erreichbare Speedup also unabhängig von der Prozessorzahl p beschränkt. Das Gesetz von Gustafson-Barsis liefert dagegen *keine* von p unabhängige obere Schranke.

Erklären Sie, wie es zu dieser Diskrepanz kommt.