timon.kelter [☺] tu-dortmund.de

Übung zur Vorlesung
Rechnerarchitektur
Sommersemester 2014

Übungsblatt 8

(10 Punkte)

Besprechung am Montag, 16. Juni 2014

8.1 Parallele Programmausführung (2 Punkte)

Warum wird durch den Einsatz von vier Prozessorkernen statt eines Prozessorkerns für die Ausführung eines Programms praktisch nie ein Speed-Up von 4 erreicht? Nennen Sie mindestens zwei Gründe.

8.2 Speed-Up (2 Punkte)

Ein sequentielles Programm $\mathcal{P}$ lässt sich in 5 Bereiche A bis E unterteilen, die aufgrund von Abhängigkeiten in dieser Reihenfolge ausgeführt werden müssen. Der Anteil der Bereiche an der Laufzeit ist in Tabelle 1 angegeben. Die Bereiche A, C und E lassen sich nicht parallelisieren. Bereich B kann in maximal 4 parallel ausführbare Einheiten transformiert werden. Für die Parallelisierbarkeit von Bereich D gibt es keine Beschränkung.

a) Wie viele Prozessorkerne werden benötigt, um einen Speed-Up von mindestens 4 zu erhalten? Geben Sie an, welche Gesetzmäßigkeit Sie angewendet haben.

   Hinweis: Beachten Sie bei der Rechnung die Beschränkung der Parallelisierbarkeit von B.

b) Ausgehend von der Lösung in Teil a) soll in Bereich D nun ein doppelt so großes Problem gelöst werden. Die Gesamtlaufzeit des parallelisierten Programms soll sich hierbei nicht ändern. Benutzen Sie Gustavsons Gesetz um den resultierenden Speed-Up *bezüglich der Gesamtlaufzeit des parallelisierten Programmes aus Teil a)* zu berechnen.

Bereich	A	B	C	D	E
Laufzeitanteil	2%	20%	5%	70%	3%

Tabelle 1: Anteile der Bereiche an der Programmlaufzeit

8.3 PRAM-Algorithmen (3 Punkte)

In der Vorlesung wurde die *PRAM-Maschine* als Grundlage zur Beurteilung der Effizienz von parallelen Algorithmen vorgestellt. Lösen Sie unter Benutzung dieses Maschinenmodells die beiden folgenden Aufgaben. Richten Sie Sich bei der Notation von Algorithmen nach der in der Vorlesung eingeführten Syntax (mit `shared` und `parallel`).

a) Das Problem das Maximum von n Zahlen zu bestimmen, ist, wie in der Vorlesung erwähnt, auf einer *CRCW-PRAM* (Concurrent Read, Concurrent Write) in $O(1)$ Schritten lösbar. Geben Sie an, wieviele Schritte man mindestens zusätzlich benötigt, wenn die zugrundeliegende Maschine eine *EREW-PRAM* (Exclusive Read, Exclusive Write) ist. Gehen Sie für die Realisierung mit der EREW-PRAM davon aus, dass 2 gleichzeitige Zugriffe das Programm zum Absturz bringen. Nutzen Sie für Ihre Lösung den Unterschied zwischen parallelen Schleifen (`parallel for`) und sequentiellen Schleifen (`for`). Außerdem können Sie in Ihrem Code die Nummer des ausführenden Prozessors $i \in (-\infty, \infty)$ über das Symbol # bestimmen.

b) Bestimmen Sie, ausgehend von den bekannten Verfahren zur Bestimmung des Maximums und der Summe von n Zahlen, einen Algorithmus, der auf einer CRCW-PRAM in $O(\log n)$ Schritten den *Rang* jeder Zahl der n Zahlen bestimmt. Der Rang einer Zahl i aus der Zahlenmenge I mit $|I| = n$ ist dabei definiert als:

$$\text{rang}(i) = |\{j | j \in I \wedge j > i\}|$$

8.4 Write-through/-back Caches (3 Punkte)

In Abb. 1 ist ein Multiprozessorsystem mit gemeinsamem Speicher und lokalen *write-through* Caches ohne Cache Kohärenzprotokoll dargestellt. Zu Beginn enthält nur der Speicher den Wert der Variable *X*. Führen Sie in Gedanken die folgende Aktionssequenz durch und beantworten Sie anschließend die Fragen a) – c).

a. Cache C1 lädt den Wert von *X* aus dem Speicher.

b. Cache C3 lädt den Wert von *X* aus dem Speicher.

c. Prozessor P3 schreibt den Wert 4 für *X* in seinen Cache (C3).

d. Prozessor P1 liest den Wert von *X* aus seinem Cache (C1).

e. Prozessor P1 schreibt den Wert 8 für *X* in seinen Cache (C1).

Fragen

a) Welchen Wert für *X* erhält Prozessor P2 durch eine Leseoperation (Aktion 6)?

b) Beantworten Sie Frage a) für *write-back* Caches. Begründen Sie Ihre Antwort.

c) Nehmen Sie an, es handelt sich um *write-back* Caches. Die *write-back*-Operationen von P3 und P1 für Variable *X* werden vor der Leseoperation von P2 (Aktion 6) ausgeführt. Außerdem wird die *write-back*-Operation von P1 für Variable *X* vor der *write-back*-Operation von P3 ausgeführt.

Welchen Wert für *X* erhält Prozessor P2 durch eine Leseoperation (Aktion 6)?

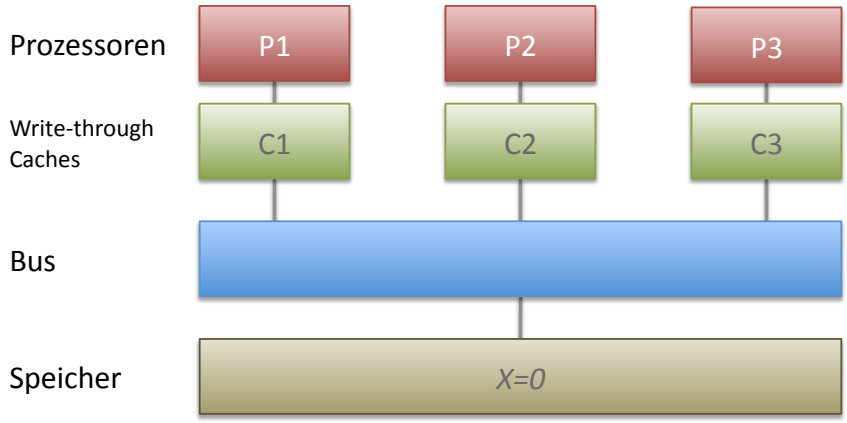

Abbildung 1: Ausgangssituation für Aufgabe 1, 2 und 3.

Allgemeine Hinweise: Die Übungstermine und weitere Informationen finden Sie unter http://ls12-www.cs.tu-dortmund.de/daes/de/lehre/lehrveranstaltungen/sommersemester-2014/rechnerarchitektur.html. Die Übungszettel werden zum Semesterbeginn online gestellt und sollen eigenständig bis zum jeweiligen Stichtag gelöst werden. Die Lösungen werden in den Gruppen besprochen. Auf Wunsch kann für diese Veranstaltung ein Übungsschein ausgestellt werden. Hierzu müssen die selbst erstellten Lösungen jeweils vor der Besprechung der Aufgaben beim Übungsgruppenleiter abgegeben werden. Dabei müssen 45% der Gesamtpunkte bei den Übungszetteln erreicht und eigene Lösungen in der Übungsgruppe präsentiert werden. Für die Teilnahme an der Klausur nach BPO 2013 / der Fachprüfung nach DPO 2001 ist der Übungsschein *nicht* erforderlich.