Universität Heidelberg

Parallel Computer Architecture

Exercise 4

Barley, Daniel
Barth, Alexander
Nisblé, Patrick

Due date: 2019-11-29, 14:00
Group 04

4.1 Numerische Integration revisited

4.1.1 Parallele Implementierung mittels PThreads

@todo: reason

4.1.2 Experimente und Evaluation

a.

Tabelle 1: Ausführungszeit $t_{compute}$ (s)

Threads \ n	2	3	4	5	6
1					
2					
4					
8					
16					
32					

Tabelle 2: Ausführungszeit $t_{compute}$ (s)

Threads \ n	2	3	4	5	6
1					
2					
4					
8					
16					
32					

b.

@todo: error

c.

@todo: discuss

4.2 Prozesse vs. Threads

a.

Prozesse liefern die Ressourcen, welche für die Ausführung eines Programms erforderlich sind.Ein Prozess hat einen virtuellen Adressraum, ausführbaren Code, offene Handles zu Systemobjekten, einen eindeutigen Prozessidentifier, eine Prioritätsklasse, minimaler und maximaler Arbeitsbedarf und mindestens einen Ausführungsthread. Jeder Prozess startet mit einem einzelnen Thread und kann zusätzliche Threads erstellen.

Ein Thread ist ein Objekt innerhalb eines Prozesses, das zeitlich festgelegt ausgeführt werden kann. Alle Threads eines Prozesses teilen sich den virtuellen Adressraum und Systemressourcen, besitzen Scheduling Prioritäten, lokalen Speicher, einen eindeutigen Threadidentifier und einen Strukturensatz, der den Thread Kontext speichert, bis die Ausführung des Threads festgelegt ist. Der Kontext beinhaltet den Maschinenregistersatz des Threads, den Kernel-Stack, einen Umgebungsblock und einen Benutzer-Stack im Adressraum des übergeordneten Prozesses.

b.

Die Kommunikation zwischen Threads ist programmiertechnisch einfacher als die zwischen mehrerer Prozesse. Kontextwechsel zwischen Threads sind schneller als Prozesswechsel. Das Betriebssystem kann Threads schneller stoppen und einen anderen starten, als mit zwei Prozessen.

4.3 Klassifikation nach Flynn

a.

Die Zuordnung zur Klasse der MISD-Systeme ist schwierig, da mit einem Datensatz mehrere Funktionseinheiten unterschiedliche Operationen durchführen. Genau genommen unterscheiden sich die Daten somit nach der Durchführung. Des Weiteren sind sie weniger verbreitet als MIMD- und SIMD-Systeme, welche geeigneter sind für übliche parallele Datentechniken.

b.

Ein Vektorrechner bearbeitet quasi-parallel mehrere Daten durch Pipelining. Dabei werden Maschinenbefehle in Teilaufgaben zerlegt. Diese Teilaufgaben werden für mehrere Befehle parallel ausgeführt.

Beim Feldrechner berechnen mehrere Recheneinheiten parallel die gleiche Operation auf verschiedenen Daten.