PFP-Team
pfp@i2.cs.fau.de

Übungsblatt 7

Abgabe bis 19.06.2019, 18:00 Uhr

Aufgabe 7.1: Lastverteilung

a) Was versteht man unter Lastverteilung?

b) Welche unterschiedlichen Ansätze gibt es?

Aufgabe 7.2: Fließband

a) Wie funktioniert die Parallelisierung mittels Fließband?

b) Wovon hängt die Skalierbarkeit bei diesem Verfahren ab?

c) Wie lange dauert es mindestens, mit einem Fließband mit 5 Stationen (mit 4s, 3s, 5s, 4s und 5s Rechenzeit) 20 Arbeitspakete fertig zu stellen? Wie hoch ist der Speed-up im Vergleich zu einer rein sequentiellen Berechnung aller Stationen?

d) Wie hoch ist der maximale Speed-up eines optimalen n-stufigen Fließbands?

Aufgabe 7.3: Teile-und-Herrsche

a) Wie sieht das Teile-und-Herrsche-Prinzip aus?

b) Welche potentiellen parallelen Aktionen gibt es?

c) Wodurch ergeben sich Probleme bei der Parallelisierung?

d) Mit welchen Methoden kann man diese Parallelisierungsprobleme in den Griff bekommen?

Bonusaufgabe 7.4: LCS (Longest Common Subsequence) **6 Punkte**

In dieser Aufgabe sollen Sie die/eine längste gemeinsame Unterfolge zweier Zeichenfolgen bestimmen. Eine Unterfolge entsteht durch beliebiges Löschen einzelner Zeichen, wobei die Reihenfolge der erhaltenen Zeichen gleich bleiben muss. Gesucht ist die längste Zeichenfolge, die Unterfolge zweier gegebener Zeichenfolgen ist. Dies lässt sich effizient mittels Dynamischer Programmierung (DP) lösen, indem Zwischenergebnisse in einer Tabelle abgelegt werden.

Details: http://en.wikipedia.org/wiki/Longest_common_subsequence_problem

a) Implementieren Sie in der abstrakten Klasse `LCS` die Methode `computeValue`, die den Eintrag (`row`, `column`) der LCS-Matrix `matrix` mittels Dynamischer Programmierung berechnet und in dieser speichert. Dafür wird der Methode die **teilweise befüllte** Matrix `matrix` und die zu untersuchenden Wörter `x` und `y` übergeben.

b) Legen Sie eine Klasse `LCSSeq` an, die von `LCS` erbt und in der Methode `longestCommonSubsequence` für die beiden Wörter `x` und `y` die LCS-Matrix sequentiell berechnet. Lesen Sie das Ergebnis anschließend mit `readMatrix` aus.

c) Das sequentielle Befüllen der Matrix lässt sich mithilfe einer Fließband-Parallelisierung verteilen. Hierbei befüllt jede Fließbandstufe eine gewisse Anzahl an Spalten. Jede Stufe muss aber aufgrund der Datenabhängigkeiten innerhalb der Tabelle warten, bis die Vorgängerstufe die benötigten Werte je Zeile berechnet hat. Dies lässt sich einfach durch die Klasse `Exchanger` lösen und soll in der Methode `longestCommonSubsequence` einer weiteren von `LCS` abgeleiteten Klasse namens `LCSExchanger` implementiert werden.

d) Anstatt synchron die Information über das Beenden einer Zeile an den Nachfolger zu übergeben, kann dies auch asynchron mittels Pufferung in `BlockingQueue`s geschehen. Dieses Verfahren soll in der Methode `longestCommonSubsequence` einer Klasse `LCSQueue` implementiert werden.

Hinweis: Alle in den Teilaufgaben b–d erstellten Klassen sollen von `LCS` abgeleitet sein. Der Konstruktor der parallelen Varianten soll als einziges Argument die Anzahl zu verwendender Aktivitäten erhalten.

6 Punkte