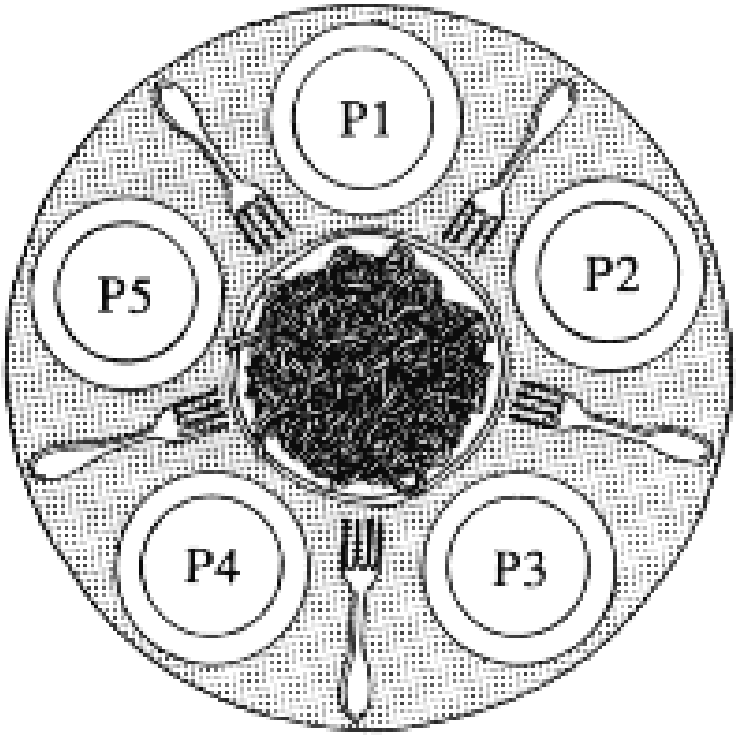
Johannes Hötzer
Informatik, Duale Hochschule Baden-Württemberg
Semester WS. 15, February 15, 2015

Übung 2. zur Vorlesung Paralleles Rechnen

- OpenMP-

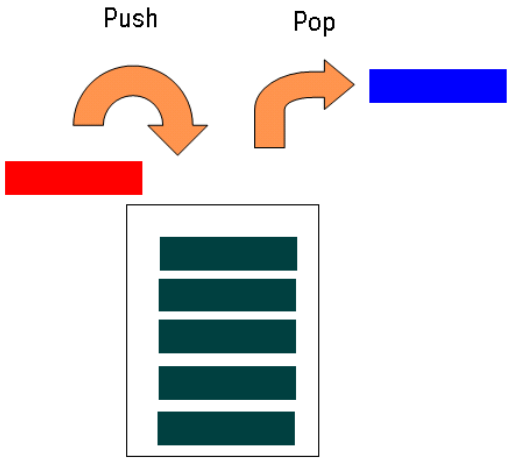
Aufgabe: 1 Die speisenden Philosophen (3 P)
Beim Problem der speisenden Philosophen geht es draum, das N Philosophen an einem runden Tisch mit N Gabeln sitzen und Essen möchten. Jedoch benötigt jeder Philosoph zwei Gabeln um essen zu können. Stellen Sie sicher das jeder Philosoph im Laufe der Zeit essen kann, d.h. er sich ohne Verklemmung sowohl mit seinem linken wie auch rechten Nachbar eine Gabel teilt.



a) Beobachtung (0.5 P) Was beobachten Sie, wenn sie das Programms ausführen?

b) Parallelisierung (2.5 P) Stellen Sie sicher das jeder Philosoph immer zwei Gabeln (seine rechte und seine linke) zum essen besitz, indem Sie locks verwenden und es nicht zu Deadlocks kommt.

Aufgabe: 2 Parallel Stack/Queue (7+5 P)
Parallele Stacks erlaube es Aufgabenpakete von einigen Threads erzeugen zu lassen und widerum von anderen Abarbeiten zu lassen, wie zum Beispiel das Herunterladen von Webseiten und das gleichzeitige Indexieren dieser.



a) Programmablauf (0.5 P) Beschreiben Sie was das Progamm, bzw. was welcher Thread tut!

b) Locking (1 P) Stellen Sie sicher, das immer nur ein Thread Zugriff auf den Stack hat, d.h. die noch leeren Funktionsrümpfe betreten kann.

c) Parallelisierung (4 P) Implementieren Sie einen parallelen Stack mit Hilfe der vorgegebenen Funktionsrümpfen!
Bei einem Stack werden immer das oberste Element entfernt bzw. neue Element oben hinzugefügt. Ist der Stack voll, bzw. leer beim Lesen, gibt der Stack 0 zurück um dem Benutzer zu signalisieren, das der Funktionsaufruf nicht erfollgreich war! Hiermit kann der Benutzer entscheiden, was das Progamm in dieser Zeit tun soll.

d) Parallelisierung (1.5 P) Stellen Sie den Stack so um, das beim Aufruf der Funktionen get/put im Falle eines leeren/vollen Stacks solange gewartet wird, bis die Operation (put, get) wider möglich ist! D.h. der Stack wird auf einen blockierenden Stack umgestellt. Das widerum heißt, das wenn der Stack leer ist, der Aufrufer solange warten, muss bis ein anderer Thread etwas auf den Stack legt. Umgekehrt muss der Thread warten, wenn er etwas darauf legen möchte, der Stack aber voll ist. Die kann mit Hilfe von aktivem testen/warten realisiert werden. Es ist nicht notwendig die Producer- und Consumer-Funktion anzupassen.

e) Optional (4 P) Stellen Sie den parallelen Stack auf eine parallele Queue um (Tipp: Java LinkedBlockingQueue).

f) Optional (1 P) Welchen Nachteil hat die Verwendung von nur einem Lock und wie könnte es besser gelösst werden?