Tweede Deeltoets Concurrency

Donderdag 1 februari 2018, 8.30 – 10.30, Educ-Γ.

Licht je antwoorden *kort* toe. Schrijf niet teveel op: onzin toegevoegd aan een goed antwoord kan aftrek geven. Vraag 3 is 2pt en de andere vragen elk 3pt. Maak vraag 1 op de voorkant, vragen 2 en 3 op pagina 2 en vragen 4 en 5 op pagina 3.

1. **Parallel Maximum:** We willen het grootste getal uit een array A van n verschillende getallen bepalen op een CRCW PRAM.

   (a) Laat zien hoe de CRCW PRAM de *conjunctie* van n condities c_i, dus $c_1 \wedge c_2 \wedge \ldots \wedge c_n$, kan bepalen in lineair (dus $O(n)$) werk en constante (dus $O(1)$) span.
   (b) Laat zien hoe het mogelijk is om het maximum van een array A van n getallen te bepalen met kwadratisch veel werk in constante span.
   (c) Geef een methode om het maximum te bepalen in $O(n\sqrt{n})$ work en constante span. **Hint:** Verdeel de invoer in $\sqrt{n}$ groepjes.

2. **Lengte van Greedy Schedule:** Een parallelle berekening met work w en span s wordt uitgevoerd op p cores volgens een Greedy Schedule. Om het aantal rondes te beredeneren, letten we speciaal op die rondes waarin *de span van de resterende berekeningsgraaf* afneemt; zulke rondes noemen we *krimpend*.

   (a) Beschrijf hoe een greedy schedule tot stand komt.
   (b) Bewijs dat er exact s krimpende rondes zijn.
   (c) Bewijs dat er hoogstens $\frac{w-s}{p}$ niet-krimpende rondes zijn en dat de lengte van de schedule begrensd is door $s + \frac{w-s}{p}$.

3. **Run-length decoding:** Leg uit (liefst met een diagram) hoe een run-length encoded string parallel *gedecodeerd* kan worden met behulp van de prefix sum. Gebruik hierbij de string: $A3B2F8D2$, die uitpakt naar $AAABBFFFFFFFFDD$. Neem aan dat de posities in de uitvoer Randomly Accessible zijn, dwz., in constante tijd geschreven kunnen worden.

4. **Parallel inproduct:** Het inproduct van vectoren A en B in $\mathbb{R}^3$ is $A_xB_x + A_yB_y + A_zB_z$. Meer algemeen is het inproduct (engels: dotproduct) van twee reeksen A en B gedefinieerd is als $\sum_{i=0}^{n-1} A_iB_i$.

   (a) Laat (eventueel met een diagram) zien hoe het inproduct in parallel kan worden berekend.
   (b) Als we voor de elementen in A en B floating point getallen gebruiken, kan het gebeuren dat de uitkomst van het parallelle algoritme niet overeenkomt met de uitkomst van een seriële berekening. Geef een voorbeeld waarbij dit gebeurt.
   (c) In de slides wordt gesteld dat voor een grote input array, parallel reduction met vector operaties de hardware vrijwel optimaal benut. Waarom is dit "vrijwel optimaal", en niet gewoon "optimaal"?

5. **QuickSort:** QuickSort verdeelt de invoer rond één gekozen waarde, de pivot, in een deel elementen kleinergelijk de pivot en een deel grotergelijk dan de pivot. Deze twee delen worden met een recursieve aanroep gesorteerd. Het verdelen, de `Partition`, is een sequentieel proces dat lineair veel werk kost.

   (a) Neem aan dat de twee delen ongeveer even groot zijn en analyseer de tijd voor sequentiele QuickSort; dwz., geef een recurrente betrekking en los die op.
   (b) Analyseer de *span* van QuickSort wanneer de twee recursieve aanroepen parallel worden gedaan.
   (c) Het is mogelijk, `Partition` te parallelliseren tot $O(\lg n)$ span. Analyseer de Span van QuickSort als deze `Partition` wordt gebruikt.