Tweede deeltoets Concurrency

8 november 2013, 11.00 – 13.00, Educ-β.

Motiveer je antwoorden *kort*! Zet je mobiel uit. Stel geen vragen over deze toets; als je een vraag niet duidelijk vindt, schrijf dan op hoe je de vraag interpreteert en beantwoord de vraag zoals je hem begrijpt. Te behalen 18pt, cijfer T2 is totaal plus 1 gedeeld door 1,7 (max 10).

1. **Fouttolerante Convergentie:** Tien stations hebben een integer invoer x_i, waarbij bekend is dat de *spread* (verschil tussen grootste en kleinste) hoogstens 70 is. Door het nemen van gemiddelden willen we de spread verkleinen, maar wel waarden krijgen tussen het oorspronkelijke minimum en maximum. Het moet asynchroon zijn en ook werken als er een of twee stations crashen. Aanpak: de stations shouten hun x, collecten er 8, en berekenen elk het gemiddelde en kappen af op een integer.
   (a) Bewijs dat het resultaat een spread van maximaal 18 heeft.
   (b) Kun je deze uitwisseling herhalen voor een nog kleinere spread? Wat wordt de spread?
   (c) Wat is de kleinste spread die door herhaling van deze handelingen mogelijk is?

2. **TaS-lock:** Een TaS-lock beschermt kritieke code.
   (a) Hoeveel test-and-set bits zijn nodig om een lock te maken voor n threads? Geef de code.
   (b) Noem enkele nadelen van het TaS-lock? Welk nadeel wordt opgeheven door het TTaS-lock, en hoe?

3. **Goed en slecht Greedy Schedule:** De lengte van een Greedy Schedule kan variëren, afhankelijk van welke ready stappen de scheduler kiest. Geef een voorbeeld van een berekeningsgraaf en twee Greedy Schedules, elk op drie cores, waarbij het ene schedule minstens anderhalf keer zo lang duurt als het andere.

4. **Pulse en PulseAll:** Wat doet de methode `Pulse` (in Java heet deze: `Notify`)? Wat is het verschil tussen `Pulse` en `PulseAll`?

5. **BenOrs tijdcomplexiteit:** In het Consensus algoritme van Ben-Or kiest een station dat geen PROPOSE ontvangt, een random bit als stem voor de volgende ronde.
   (a) Bewijs dat na elke ronde de kans minstens $\frac{1}{2^{n-1}}$ is dat alle stations *dezelfde* waarde als stem hebben.
   (b) Geef een zo nauwkeurig mogelijke schatting voor het verwachte aantal berichten (in een run zonder crashes).

6. **Parallel MaxOneRow:** De *MaxOneRow* van een rij bits is het maximale aantal aaneengesloten 1-en; bv van 01011001111100111110 is de MaxOneRow 5 (zie posities 7 en 14). Je kunt de MaxOneRow berekenen met een recursieve module die van een rij bits deze drie waarden oplevert: `p`: het aantal 1-en waarmee de rij begint; `i`: de MaxOneRow; `s`: het aantal 1-en waar de rij op eindigt.
   (a) Wat is de beste sequentiele tijd om de MaxOneRow te berekenen?
   (b) Hoe vind je de `p`, `i` en `s` van een rij uit die van de linker- en rechterhelft?
   (c) Analyseer de *work* van het resulterende algoritme.
   (d) Analyseer de *span* van het resulterende algoritme.
   (e) Is het parallelle algoritme *efficient* en is het *optimaal*? Leg uit!