Eerste Hertoets Concurrency

24 maart 2016, 11.00–13.00, Unnik-220.

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. **Cijfer:** Maak vraag 1 en 2 op pagina 1, 3 en 4 op pagina 2, en 5 en 6 op pagina 3. Vraag 1, 5 en 6 zijn 3pt, vragen 3 en 4 elk 2pt, vraag 2 is 4pt. Cijfer T1 is totaal plus 1, gedeeld door 1,7.

1. **Amdahl and friends:** Een programma besteedt 5 seconden aan werk dat niet parallel uitgevoerd kan worden en 10 seconden aan werk dat wel parallel uitgevoerd kan worden.

   (a) Wat is volgens Amdahl's Law de maximaal haalbare versnelling van dit programma op een quadcore CPU? Laat hyperthreading buiten beschouwing.
   (b) Waarom kan dit volgens Gustafson en Barsis toch beter?
   (c) Welke informatie is nodig om met het work-span model een betere inschatting te maken van de te verwachten prestatie van een parallelle variant van het programma?

2. **Hardware:** (a) Wat is de relatie tussen speculative execution en branch prediction?
   (b) Wat is false sharing, en hoe beïnvloedt dit de performance van een parallel programma?

3. **Vectorizatie:** Twee vormen van paralellisme zijn instruction level paralellism en thread level paralellism.

   (a) Wat is het verschil tussen beide vormen?
   (b) Wat is het verschil tussen een thread en een stream (ook wel lane)?

4. **Fair executie:** In dit programma kijkt thread 1 steeds naar `w`, die door thread 2 afwisselend op `true` en `false` wordt gezet. Ga uit van read/write atomicity; initieel is `w=s=True` en `t=0`:

```
Thread 1:          Thread 2:
  while (w)          while (s)
  { t = t + 1 }        { w = !w }
  s = False           print t
```

   Beschrijf een oneindige executie die mogelijk is onder een fair scheduler.

5. **LockOne:** Hier staan de lock en unlock van de LockOne klasse (voor thread i).

```
public void lock()            public void unlock()
{ flag[i] = true;                  { flag[i] = false ; }
  while (flag[j]) {}  }
```

   (a) Aan welke drie eisen moet een lock implementatie voldoen?
   (b) Welke van deze eisen is/zijn voor LockOne niet voldaan? Waarom?
   (c) Als je de twee regels in `lock` verwisselt, is dan het probleem opgelost?

6. **Multivalued register:** Je kunt een m-waardig *regular* register maken uit een array van m regular bits. De Writer schrijft waarde x door bit x op 1 te zetten en de lagere bits op 0, de reader zoekt vanaf positie 0 naar de eerste 1:

```
Write(x):                    Read:
  r[x].write(1)                for(i=0; i<m; i++)
  for (i=x-1; i>=0; i--)         if (r[i].read == 1) return i;
    r[i].write(0)              // A
```

(a) Laat zien dat de Reader kan falen wanneer de Writer ook de hogere bits op 0 zet (met een loopje `for (i=x+1; i<m; i++) r[i].write(0)`).
(b) Is het gebouwde register ook atomic? Leg uit!
(c) Bobs compiler weigert de `Read` code wegens executiepaden zonder `return`. Daarom wil Bob als default op plek `A` een `return ...` zetten; met welke waarde achter de `return` is de methode correct?