Multithreading sudoku

Matisse Bonvin
Benjamin Fraeyman

Vak: Computersystemen

25 mei 2018

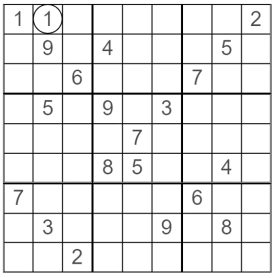
Inhoudsopgave

Inleiding

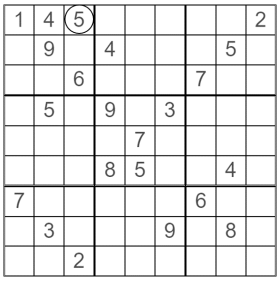
Een sudoku kan op verschillende manieren worden opgelost, de populairste is door middel van brute force. Er zijn veel strategieën om een sudoku op te lossen, oplossingen kunnen soms niet uniek zijn of zelfs onbestaand. Als de sudoku door een computer moet worden berekend, is de makkelijkste manier brute force. Dit heeft 2 voordelen, het programma kan weergeven dat het sudoku onoplosbaar is of het programma kan meerdere oplossingen weergeven.

1 Algoritme

Vooraleer er parallelmogelijkheden overlopen worden, wordt eerst het algoritme beschreven om één sudoku op te lossen.
Het algoritme overloopt alle cellen van de sudoku. Als het vakje het meest oplosbaar zou zijn (er zijn bijvoorbeeld veel cijfers in bijhorende kolom en rij) zal het algoritme de cel tijdelijk vervangen door een cijfer van 1 tot 9. Daarna wordt er gekeken of het cijfer wel past in de cel door de bijhorende rij, kolom en 3x3 vak te controleren.

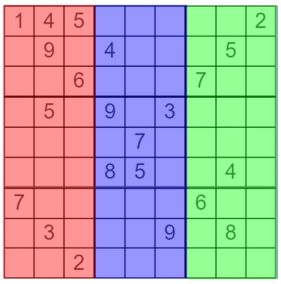


Indien dit cijfer voorkomt in één van de drie gevallen, zal er een nieuwe cijfer worden getest. Dit proces herhaalt zich totdat het cijfer zou kunnen passen. Daarna komt de volgende lege cel. Dat is dan, om het algoritme een beetje te optimaliseren, de voorgaande geraden cijfer plus één. Als een cijfer niet mogelijk is, wordt er weer 1 plaats achteruit gegaan en opnieuw geprobeerd totdat alle cijfers zijn ingevuld. Er wordt enkel gebruik gemaakt van oplosbare sudoku's zodat er geen zorgen hoeft worden gemaakt omtrent onoplosbaarheid.



2 Strategieën

Nu moet er nog een manier bedacht worden om sudoku's op de één of andere manier te verdelen over CPU's.
Een strategie zou zijn om threads te verdelen over de rijen, kolommen en/of subvakken van een sudoku. Figuur 1 geeft weer hoe bijvoorbeeld 3 threads verdeeld zouden zijn. Echter is hier het probleem dat het programma zeer ingewikkeld wordt doordat alle threads afhankelijk zijn van elkaar. Als een cijfer in een kolom van de eerste thread verandert zal een andere thread van een andere kolom daar rekening mee moeten houden.



Figuur 1: 3 threads verdeeld over de kolommen, elke thread is aangeduid met een kleur (strategie 1)

Een tweede mogelijkheid is om op het einde threads te kunnen gebruiken als controle van de sudoku. Figuur 2 toont hoe 9 threads verdeeld zouden kunnen zijn. De threads zouden verdeeld kunnen zijn over de aantal rijen, kolommen en/of subvakken. In dit geval zou dit onnodig werk zijn aangezien elke stap in het algoritme al controleert of het cijfer past. De sudoku is op het einde altijd al juist opgelost.

Figuur 2: 9 threads verdeeld over aantal subvakken, elke thread is aangeduid met een kleur (strategie 2)

De laatste mogelijkheid is om veel sudoku's over een aantal threads te verdelen. Dit is dan ook de strategie die werd uitgewerkt en verder in dit verslag wordt beschreven.

3 Multi-threading

Om het programma parallel te implementeren, worden threads gebruikt als solvers. Indien er dus meer threads zijn, dan kunnen er meer sudoku's per instantie opgelost worden. Om de workload te verdelen wordt er gebruik gemaakt van een executor service. Deze service bevat een fixed threadpool, dit is een pool van threads die op voorhand een grootte gekregen heeft. Deze threadpool dient om het aantal threads bij te houden. Indien een thread zijn werk gedaan heeft komt deze in de pool. Indien de executorservice werk klaar heeft staan, deelt hij deze uit aan de beschikbare threads uit de pool.

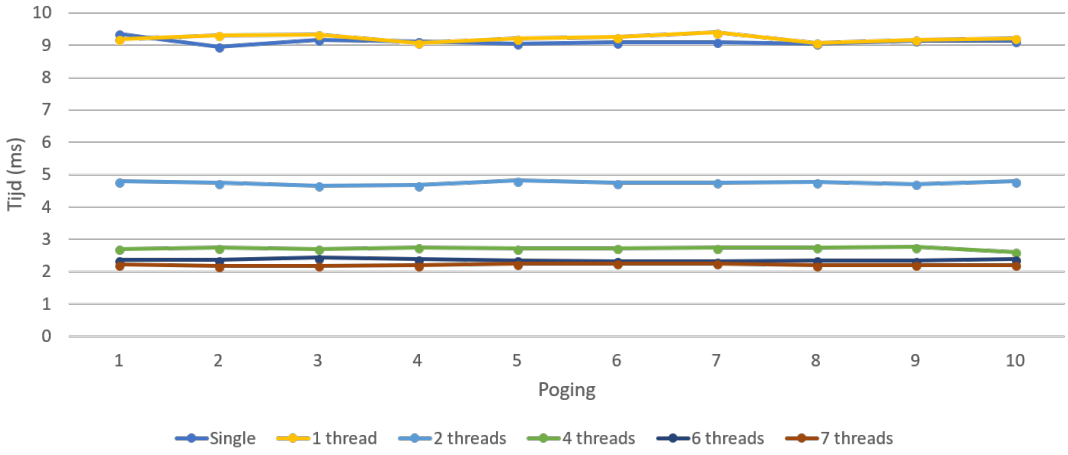
Bij het programma werkt het parallel deel tot maximaal 'aantal CPU-threads - 1', dit is omdat er dan altijd 1 thread over is voor het OS en de executorservice.

4 Uitvoer programma

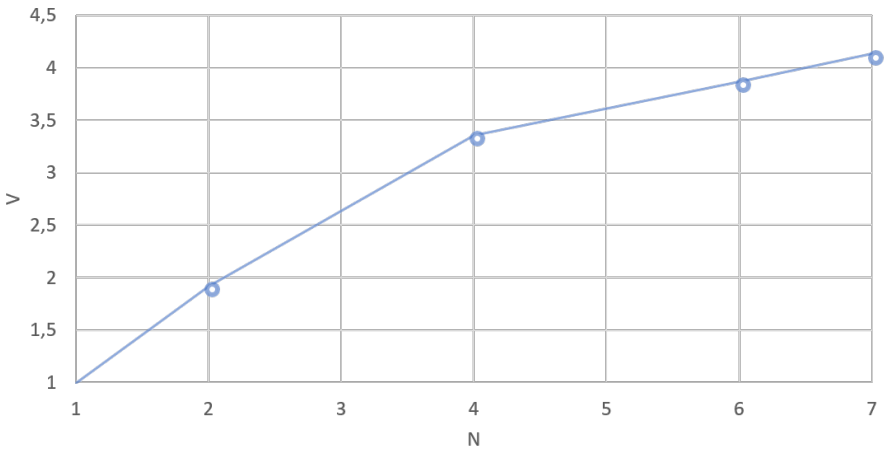
Hier wordt de executie van het programma uitgelegd.

1. Keuze tussen basisprogramma of het parallel aangepaste programma (hier wordt de stresstest mode uitgelegd, deze overloopt beide implementaties en gaat bij het parallel deel van 1 tot max aantal threads).
2. Inladen van de sudoku's uit een file die op storage staan.
3. Sequentieel alle sudoku's oplossen door middel van de eerdere uitgelegde regels.
4. Vervolgens wordt dit opnieuw met de parallelle implementatie gedaan.
5. Executor service wordt aangemaakt met x threads.
6. Executor service deelt sudoku's toe aan de threads en vermindert telkens het aantal beschikbare threads met 1.
7. Wanneer er geen threads meer beschikbaar zijn, wacht de executor service.
8. Van zodra een thread klaar is met zijn sudoku, gaat deze terug in de threadpool.
9. Het doorlopen is klaar van zodra alle sudoku's opgelost zijn.

5 Resultaten



Figuur 3: Prestatieverbetering bij meerdere CPU's



Figuur 4: Versnellingscurve

```
------------ Loading ------------
Loading sudokus into memory completed:
    - Loading took 4,2 ms
------------ Solving ------------
------------ Sequential ------------
Solving 1000 puzzles - Sequential
Solving completed!
    - Solving took 9,15 seconds
    - Time per puzzle: 9,15 ms
Verifying results:
    - No unsolvable sudokus or incorrect solutions!

------------ Parallel ------------
    This cpu has 8 threads, we will run up to: 7 threads!
    Solving 1000 puzzles - Parallel

Running in parallel with 1 threads:
    Solving completed!
        - Solving took 8,92 seconds
        - Time per puzzle: 8,92 ms
    Verifying results:
        - No unsolvable sudokus or incorrect solutions!

Running in parallel with 2 threads:
    Solving completed!
        - Solving took 4,57 seconds
        - Time per puzzle: 4,57 ms
    Verifying results:
        - No unsolvable sudokus or incorrect solutions!

Running in parallel with 3 threads:
    Solving completed!
        - Solving took 3,16 seconds
        - Time per puzzle: 3,16 ms
    Verifying results:
        - No unsolvable sudokus or incorrect solutions!

Running in parallel with 4 threads:
    Solving completed!
        - Solving took 2,66 seconds
        - Time per puzzle: 2,66 ms
    Verifying results:
        - No unsolvable sudokus or incorrect solutions!

Running in parallel with 5 threads:
    Solving completed!
        - Solving took 2,47 seconds
        - Time per puzzle: 2,47 ms
    Verifying results:
        - No unsolvable sudokus or incorrect solutions!
```

Figuur 5: Output

Besluit

Er is een duidelijke versnelling merkbaar wanneer men meerdere cores aan het werk zet. De versnelling is niet oneindig en vlakt duidelijk af naarmate men meer cores gebruikt.
De grootste versnelling is er bij overgang van 1 naar 2 cores.
Met behulp van "Amdahls Law"werd berekend dat het serieel deel zo'n 7 - 10% beslaat van het programma.
Een extra verbetering kan zijn: sudoku's inladen en terwijl al oplossen.

Dit was een leerzaam labo waarbij er "gesplitst"werd leren denken om het probleem op te lossen.