# PDS
# OpenMP Convolutie

Randy Thiemann

20 oktober 2014

## 1 Aanpak

Ik heb voor de implementatie twee plaatsen gevonden om multi-threading toe te passen.

- In de binnenste for loops waar de hadamar operatie van de afbeelding en de kernel wordt uitgevoerd.
- In de buitenste for loops waar de convolutiefunctie wordt aangeroepen.

Voor het programma te testen op het VSC, heb ik beide methodes op mijn persoonlijke computer getest. Hier uit is gebleken dat de eerste methode veel minder effectief is dan de tweede methode, de eerste methode is 4 tot 5 keer trager afhankelijk van de scheduler, en dat als we deze combineren de code zelfs trager draait dan zonder enige OpenMP optimizatie.

Ik heb dus besloten enkel de tweede methode toe te passen. Op mijn computer, met een 7216x5412 afbeelding en een 31x31 kernel, duurt het gemiddeld 44 seconden op zowel static, dynamic als guided scheduling.

## 2 Resultaten

Op het VSC heb ik met dezelfde afbeelding en kernel 10 runs gedaan met drie verschillende schedulers: static, dynamic en guided. De resultaten zijn als volgt:

De dynamische runtimes in seconden waren: 14.46, 14.60, 14.53, 14.47, 15.95, 14.58, 14.52, 15.81, 14.52 en 15.15. Dit geeft een gemiddelde runtime van 14.86 seconden en een standaardafwijking van 0.575 seconden.

De statische runtimes in seconden waren: 14.97, 14.45, 14.47, 14.98, 14.45, 14.47, 15.09, 15.09, 14.97 en 16.84. Dit geeft een gemiddelde runtime van 14.98 seconden en een standaardafwijking van 0.712 seconden.

De guided runtimes in seconden waren: 14.63, 14.43, 14.43, 14.46, 15.67, 15.88, 14.45, 15.84, 14.99 en 16.48. Dit geeft een gemiddelde runtime van 15.13 seconden en een standaardafwijking van 0.77 seconden.

Het valt hier op dat dynamische scheduling gemiddeld sneller gaat, en dat deze de kleinste standaardafwijking heeft. Voor deze specifieke implementatie is dynamische scheduling dus veruit de beste.