Parallellisme - Project report

Hans Van der Ougstraete

April 10, 2020

Contents

1 Implementatie

1.1 Fase 1

Fase 1 is opgedeeld in twee delen, het eerste deel maakt een lijst met relevante businesses door in de base case de evaluaterevalence methode uit SequentialSearch aan te roepen. De Yelpdata wordt dus in stukken opgedeeld tot het kleinste deel een business is. Deze lijst wordt in het tweede deel aan de hand van een parallele mergesort gesorteerd. Deze code is terug te vinden in de ParallelSearch klasse.

1.2 Fase 2

In progress

2 Evaluatie

De benchmarks op eigen computer (quad core i5-2300 @2.80GHz) zijn 3 maal uitgevoerd. Voor SequentialSearch, ParallelSearch met 2 workers en ParallelSearch met 4 workers. Dit telkens 1500 keer. Het weergeven van de resultaten is per preset zo kan men vergelijken tussen de drie verschillende benchmarks voor 1 bepaalde preset. Hier bespreek ik de resultaten en manier van werken in het algemeen. Ik denk dat ik ga kiezen voor een boxplot per preset voor seq, 2p en 4p.

2.1 Fase 1

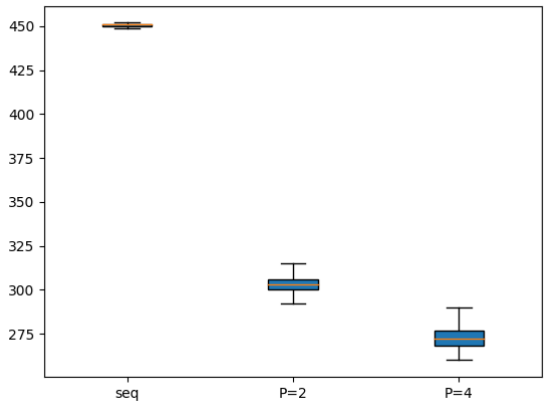

Figure 1: Preset1

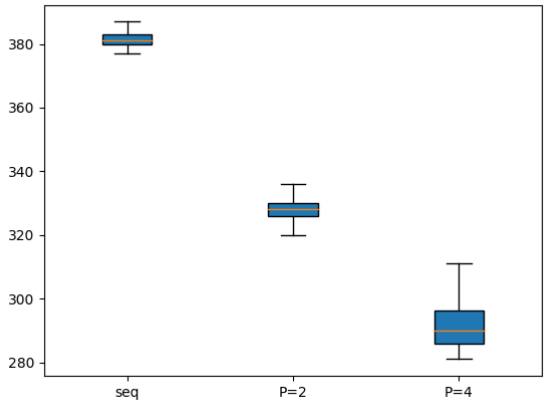

Figure 2: Preset2

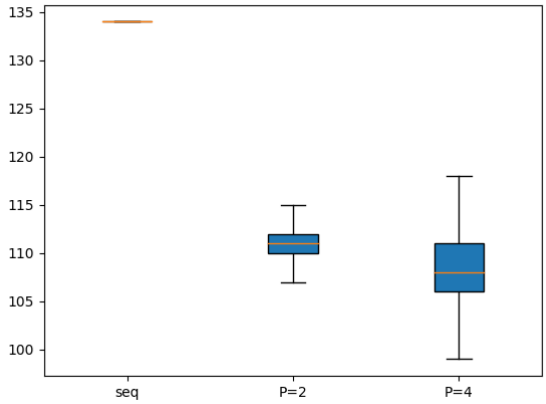

Figure 3: Preset3

2.1.1 Overhead

Gegeven dat de overhead per preset x is kunnen we de efficientie bepalen van de parallele implementatie. De overhead van het verdelen in threads in niet aanwezig bij de sequentiele versie omdat de sequentiele versie niet opdeeld in kleine taken. De overhead zou verminderd kunnen worden door een cutoof te gebruiken. De overhead is verschillend per preset omdat het opdelen van het werk afhangt van de respectievelijke data.

2.1.2 Speed-up

Een perfecte speed-up zou gelijk zijn aan het aantal extra processors, dit is hier niet het geval... Schaalt de berekening voor meer cores?

2.2 Fase 2