Reflectie

Ik vind dat het geslaagd is om de zeef van eratosthenes te implementeren met MPI en MP. Ik vond het wel een hele lastige opdracht en heeft me dan ook aardig wat tijd gekost (een stuk meer dan normaal).

Wat er slecht ging was dat om MPI en MP toepassen ik mijn sequentieel ontwerp heb moeten aanpassen en het een heel stuk minder efficiënt moeten te maken zodat het werkte. Dus met 1 MPI en 1 proces is het totaal niet efficiënt. Maar het gebruik maken van meerde computers (MPI) laat duidelijk zien dat het een positief effect geeft. Wat erg leuk is om te zien dat dat werkt.

Het lastigste van de opdracht vond ik om de lijst/zeef te verdelen over de computers (MPI). De k's verdelen over meerdere processen (MP) vond ik een stuk makkelijker.

Pseudo code

Hij krijgt binnen hoe groot n is

Hij krijgt binnen hoeveel computers er zijn

Elke computer maakt een deellijst van de totale lijst

Elke computer doet alsof de deelijst niet bij index 0 begint maar met de index waar die eigenlijk zou moeten starten. (Behalve computer 0)

Bij n is 100 en bij 4 computers doet computer 0 deellijst 0-25, computer 1 25-50, computer 2 50-75 en de laatste computer deellijst 75-101.

Dan maakt elke computer nog het gegeven aantal processen aan.

Nu gaat de computer de k's verdelen over de processen

Elk proces kan bij de deellijst van zijn computer

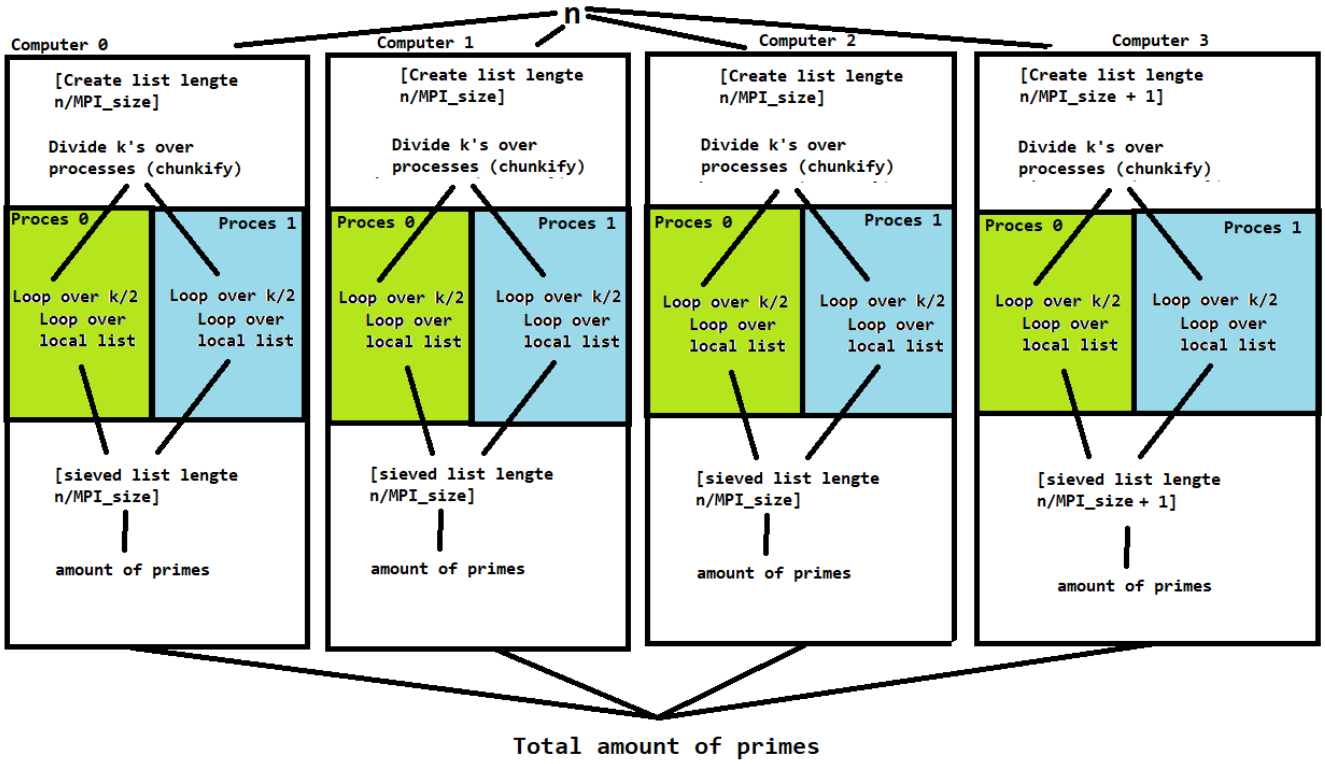
En zeeft als het waren zijn k's uit de deellijst.

Dit doen alle processen parallel over zijn eigen deellijst

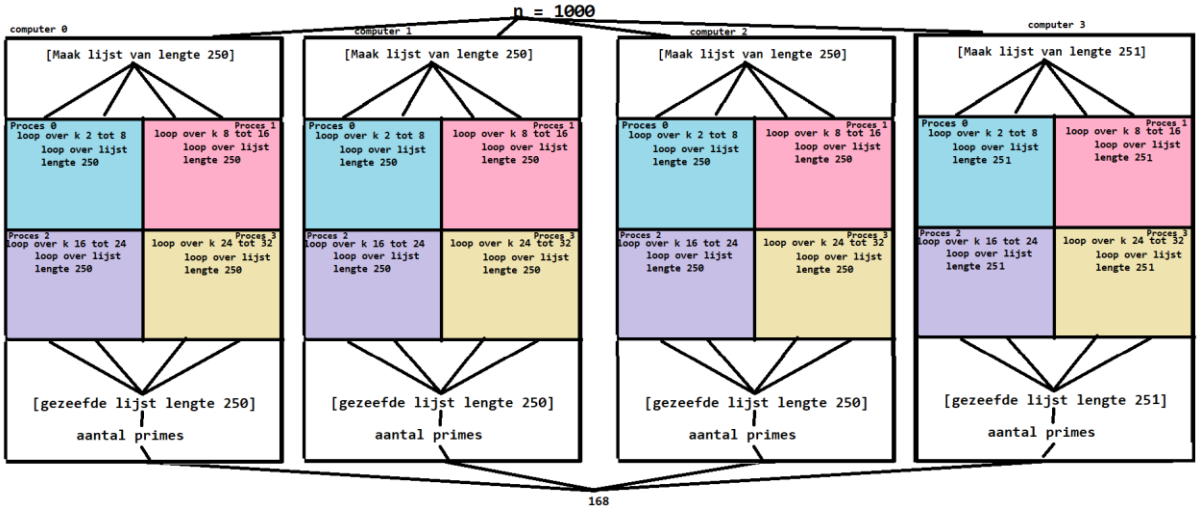
Dan berekent elke computer hoeveel priemgetallen die heeft

Dit wordt bij elkaar opgeteld

Ontwerp

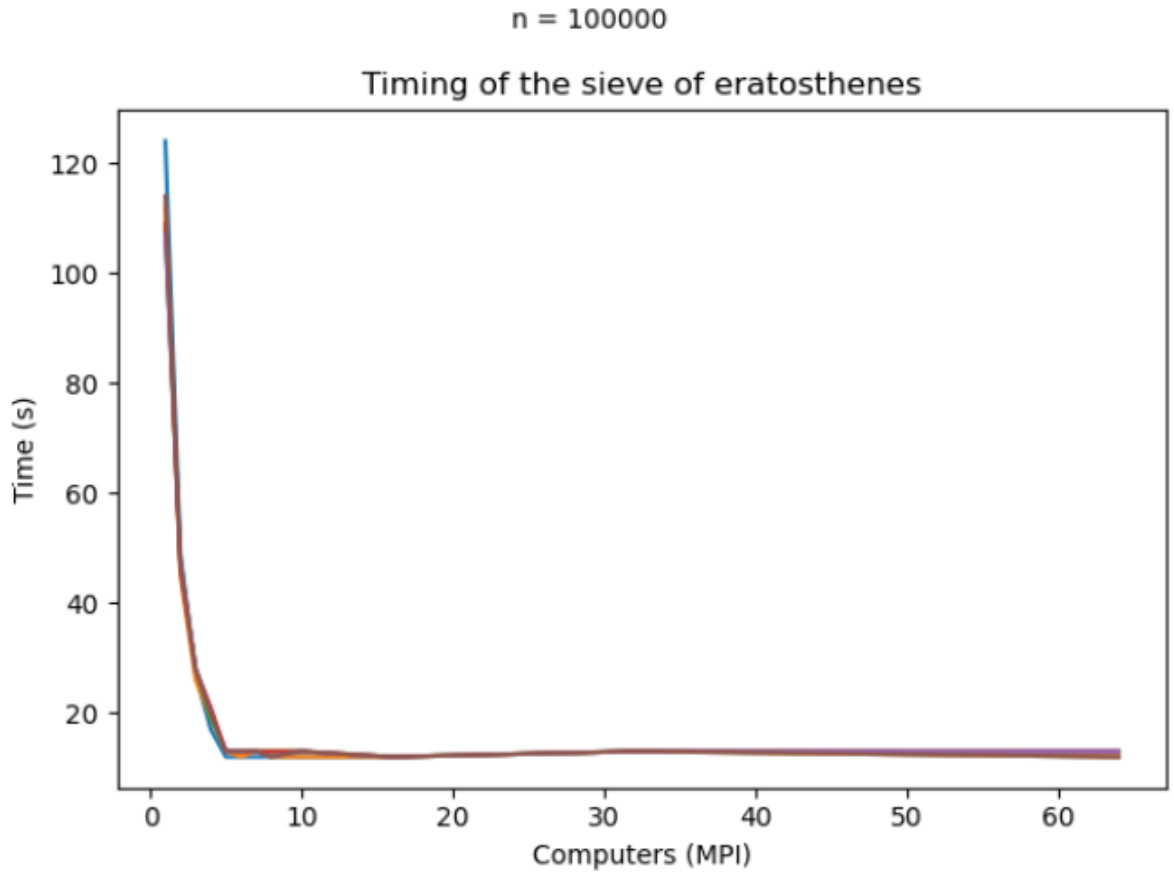


Dit is een ontwerp met 4 computers (MPI) en 2 processen (MP).



Dit is een voorbeeld met 4 computers (MPI) en 4 processen (MP). De kleurtjes zijn de processen.

Uitkomst



Er is duidelijk te zijn dat het toevoegen van computers de tijd verlaagd. Dit heeft alleen maar een effect tot en met 6 computers (MPI). Na de 6 computers blijft het even snel. Dit komt omdat de hardware dus blijkbaar maximaal 6 computer parallel kan laten lopen.

Het toevoegen van processen (MP) heeft jammer genoeg geen effect. Ik heb alles gedraaid met verschillende hoeveelheden processen en die hebben in de grafiek allemaal hun eigen kleur. Alleen lopen ze zo goed als gelijk, waardoor dit lastig te zien is. Waarom het toevoegen van meerdere processen met MP geen effect geen weet ik niet.

Sequentieel

	Tijd (s)	
n	sequentieel	sequentieel vectorized
10000	0.01	0.001
100000	0.2	0.02
1000000	2.6	0.3
10000000	30.8	3.6
100000000	360.6	36.3

Hier is ook duidelijk te zien dat de sequentiële variant. De sequentiële variant draait de zelfde n in 0.2 seconden en de sequentiële vectorized zelfs in 0.02 seconden. Overigens is de vectorized versie ongeveer 10 keer zo snel. Dit komt omdat het gebruik van vectors de complexiteit flink naar beneden haalt.

Conclusie

Mijn implementatie van de zeef van eratosthenes met gebruik van MPI en MP is duidelijk niet efficiënt. Misschien is dit wel het geval bij een n die naar oneindig gaat. Dit heb ik niet getest omdat het dan simpel weg te lang duurt. Mijn advies zou zijn: gebruik de sequentiële vectorized variant, want die is duidelijk het met efficiënt. Dit zou zelfs nog sneller kunnen met gebruik van numpy, hier heb ik overigens niet voor gekozen omdat ik zou dicht mogelijk op de standaard python code wilde blijven.