INF2440-Oblig2 Lars Haseid (larshalv)

CPU: 1.4 GHz Intel Core i5 (4 kjerner)

Tid oppgitt i ms

n	Sekvensiell	Parallell
2m	7.5841820	58.1727
20m	171.8687	147.3146
200m	2552.8552	1707.1038
2mrd	34446.82	22611.571

Kompilering av program: “javac Oblig2.java”
Bruk av program: “java -Xmx4000m Oblig2 <nummer> <metode>”
nummer = hvor høy N skal være
metode = “0” for sekvensiell eller “1” for parallell

I min besvarelse har jeg ikke brukt bit-mønstre for å løse problemet, jeg utnyttet meg av tilbudet som ble gitt i oppgaven; “Selvsagt står du helt fritt til å implementere den på en annen måte hvis du vil det”. Merket fort at å få plass til N = 2mrd var litt problematisk, løste det med å kjøre programmet med mere RAM, fikk ikke kjørt både sekvensiell og parallell på en gang, derfor metodevalget som parameter.

Når vi ser på tidtabellen ser vi at den parallelle versjonen gir speedup allerede etter N = 20m. Når N øker så øker også avstanden mellom sekvensiell og parallell.

Jeg klarte ikke å parallellisere faktoriseringen ordentlig, men jeg lagde en viss form for parallellisering, spesifikt det oppgaven spurte oss om ikke å gjøre. ville bare ha en form for parallell faktorisering for å vise forståelse. Det ble allikevel ingen speedup. Jeg klarte ikke å komme opP med en fungerende algoritme. Tankegangen var å splitte opp de primtallene jeg hadde funnet jevnt på vær tråd og så prøve å dele tallet på disse primtallene, om divisjonen ga 0 i rest kan primtallet lagres som en faktor, tallet oppdateres etter divisjonen. Slutt tallet i vær tråd må jo da bli faktorisert på et lavere nivå eller noe?. Det var der det stoppet.

God Påske :)